SOFTWARE DOCUMENTATION
THAT THING
WE LOVE TO HATE

Ludovic Courtès
Service d’expérimentation & développement
Inria Bordeaux Sud-Ouest

19 June 2012
introduction

soothing the developer’s conscience

meeting the user’s expectations

tools for doc

outro
Hello, 911? I just tried to toast some bread, and the toaster grew an arm and stabbed me in the face!

DID YOU READ THE TOASTER'S MAN PAGE FIRST?

Well, no, but all I wanted was—

CLICK

http://xkcd.com/293/
user vs. developer

Tech# 710
If you don't like my attitude call:
1-800-555-RTFM
introduction

soothing the developer’s conscience

meeting the user’s expectations

tools for doc

outro
lazy developer’s great idea #0

“Let’s get this intern to scribble a web page!”
lazy developer’s great idea #0

Win an internship with [Software!](#)

By Zara (ZaraL). Thursday, July 30, 2009 12:37:00 PM

Fully paid internship as an Online Writer with [Software](#)

Software is seeking a talented wordsmith for a fully paid six-month internship as an Online Writer
lazy developer’s great idea #1

“We’ll setup a wiki, and let users write their doc”
“We’ll setup a wiki, and let users write their doc”
“Great, and don’t forget a link to Bob’s web page.”
Documentation

A description of the options accepted by [redacted] is available [here](#).

A grammar of the SmPL language is available in [pdf](#) or [html](#). This document also contains some annotated examples.

More examples are available [here](#).

The [Wiki](#) contains more hints about using [redacted].

The LWN articles about [redacted] may provide a useful introduction: [Semantic patching with [redacted]](#) by Valerie Asensio, and [Evolutionary development of a semantic patch using [redacted]](#) by Wolfram Gang. [Clic Labourd](#) has also written an article about his [experiences](#).
lazy developer’s great idea #2

“Let’s generate doc and be done with it!”
lazy developer’s great idea #2

“Let’s generate doc and be done with it!”

“Best of all: it’ll be consistent with the code!”
lazy developer’s great idea #2

Module \texttt{libvirt}

Provides the interfaces of the \texttt{libvirt} library to handle virtualized domains

Table of Contents

Macros

```c
#define LIBVIR_VERSION_NUMBER
#define VIR_COPY_CPUMAP
#define VIR_CPU_MAPLEN
#define VIR_CPU_USABLE
#define VIR_DOMAIN_BANDWIDTH_IN_AVERAGE
#define VIR_DOMAIN_BANDWIDTH_IN_BURST
#define VIR_DOMAIN_BANDWIDTH_IN_PEAK
#define VIR_DOMAIN_BANDWIDTH_OUT_AVERAGE
#define VIR_DOMAIN_BANDWIDTH_OUT_BURST
#define VIR_DOMAIN_BANDWIDTH_OUT_PEAK
#define VIR_DOMAIN_BLKIO_DEVICE_WEIGHT
#define VIR_DOMAIN_BLKIO_FIELD_LENGTH
#define VIR_DOMAIN_BLKIO_WEIGHT
#define VIR_DOMAIN_BLOCK_IOTUNE_READ_BYTES_SEC
#define VIR_DOMAIN_BLOCK_IOTUNE_READ_IOPS_SEC
#define VIR_DOMAIN_BLOCK_IOTUNE_TOTAL_BYTES_SEC
#define VIR_DOMAIN_BLOCK_IOTUNE_TOTAL_IOPS_SEC
#define VIR_DOMAIN_BLOCK_IOTUNE_WRITE_BYTES_SEC
#define VIR_DOMAIN_BLOCK_IOTUNE_WRITE_IOPS_SEC
#define VIR_DOMAIN_BLOCK_STATS_ERRS
#define VIR_DOMAIN_BLOCK_STATS_FIELD_LENGTH
#define VIR_DOMAIN_BLOCK_STATS_FLUSH_REQ
#define VIR_DOMAIN_BLOCK_STATS_FLUSH_TOTAL_TIMES
#define VIR_DOMAIN_BLOCK_STATS_READ_BYTES
#define VIR_DOMAIN_BLOCK_STATS_READ_REQ
#define VIR_DOMAIN_BLOCK_STATS_READ_TOTAL_TIMES
```
lazy developer’s great idea #2

Types

typedef struct_virTypedParameter virBlkioParameter
typedef virBlkioParameter * virBlkioParameterPtr
typedef enum virBlkioParameterType
typedef enum virCPUCompareResult
typedef struct_virConnect virConnect
typedef struct_virConnectAuth virConnectAuth
typedef virConnectAuth * virConnectAuthPtr
typedef struct_virConnectCredential virConnectCredential
typedef virConnectCredential * virConnectCredentialPtr
typedef enum virConnectCredentialType
typedef enum virConnectDomainEventBlockJobStatus
typedef enum virConnectDomainEventDiskChangeReason
typedef enum virConnectFlags
typedef virConnect * virConnectPtr
typedef struct_virDomain virDomain
typedef struct_virDomainBlockInfo virDomainBlockInfo
typedef virDomainBlockInfo * virDomainBlockInfoPtr
typedef enum virDomainBlockJobAportFlags
typedef unsigned long long virDomainBlockJobCursor
typedef struct_virDomainBlockJobInfo virDomainBlockJobInfo
typedef virDomainBlockJobInfo * virDomainBlockJobInfoPtr
typedef enum virDomainBlockJobType
typedef enum virDomainBlockRebaseFlags
typedef enum virDomainBlockResizeFlags
typedef virDomainBlockStatsStruct * virDomainBlockStatsPtr
typedef struct_virDomainBlockStats virDomainBlockStatsStruct
typedef enum virDomainBlockedReason
typedef enum virDomainConsoleFlags
typedef struct_virDomainControlInfo virDomainControlInfo
typedef virDomainControlInfo * virDomainControlInfoPtr
typedef enum virDomainControlState
typedef enum virDomainCoreDumpFlags
typedef enum virDomainCrashedReason
typedef enum virDomainCreateFlags
## Functions

```c
typedef void (*virConnectDomainEventDiskChangeCallback)(virConnectPtr conn, int callbackID)

int virConnectDomainEventDeregisterAny (virConnectPtr conn, virConnectDomainEventCallback cb)

int virConnectDomainEventDeregister (virConnectPtr conn, virConnectDomainEventCallback cb)

int virConnectDomainEventCallback (virConnectPtr conn, virDomainPtr dom, int event, int detail, void *opaque)

typedef void (*virConnectDomainEventBlockJobCallback)(virConnectPtr conn, virDomainPtr dom, const char * disk, int type, int status, void * opaque)

int virConnectDomainEventBlockJobCallback (virConnectPtr conn, virDomainPtr dom)

void virConnectDomainEventBlockJobCallback (virConnectPtr conn, virDomainPtr dom)

int virConnectCompareCPU (virConnectPtr conn, const char ** xmlCPUs, unsigned int ncpus, unsigned int flags)

int virConnectClose (virConnectPtr conn)

int virConnectBaselineCPU (virConnectPtr conn, const char ** xmlCPUs, unsigned int ncpus, unsigned int flags)

int virConnectAuthCallbackPtr (virConnectCredentiaPIPtr cred, unsigned int ncred, void * cbdata)

int virConnectAuthCallbackPtr (virConnectPtr conn, void * cbdata)

int virConnectAuthCallbackPtr (void * cbdata)

typedef void (*virConnectDomainEventBlockJobCallback)(virConnectPtr conn, virDomainPtr dom, const char * disk, int type, int status, void * opaque)

typedef void (*virConnectDomainEventCallback)(virConnectPtr conn, virDomainPtr dom, int event, int detail, void *opaque)
```

**lazy developer’s great idea #2**

L. Courtès — Software Documentation—That Thing We Love to Hate
lazy developer’s great idea #2

Functions

typedef virConnectAuthCallbackPtr
int virConnectAuthCallbackPtr (virConnectCredentialPtr cred,
unsigned int ncred,
void * cbdata)

char * virConnectBase64DecToAscii (const char * base64,
size_t len,
const char * xmlCPUs,
const char * xmlMemory,
const char * xmlFlags)

int virConnectClose (virConnectPtr conn)

int virConnectCloseAny (virConnectPtr conn,
const char * xmlDesc,
unsigned int flags)

typedef virConnectDomainEventBlockJobCallback
void virConnectDomainEventBlockJobCallback (virConnectPtr conn,
virDomainPtr dom,
const char * disk,
int type,
int status,
void * opaque)

typedef virConnectDomainEventCallback
int virConnectDomainEventCallback (virConnectPtr conn,
virDomainPtr dom,
int event,
int detail,
void * opaque)

int virConnectDomainEventDeregister (virConnectPtr conn,
virConnectDomainEventCallback cb)

int virConnectDomainEventDeregisterAny (virConnectPtr conn,
int callbackID)

typedef virConnectDomainEventDiskChangeCallback
void virConnectDomainEventDiskChangeCallback (virConnectPtr conn,
virDomainPtr dom)
lazy developer’s great idea #2

Functions

```c
typedef virConnectAuthCallbackPtr
int virConnectAuthCallbackPtr (virConnectCredentialPtr cred,
unsigned int ncred,
void * cbdata)

char * virConnectBase64DecodeUrl(char * encoded,
const char * base64,
unsigned int len)

int virConnectBase64EncodeUrl(char * dest,
const char * src,
unsigned int len)

typedef virConnectDomainEventBlob
void virConnectDomainEventBlob (connectPtr conn,
const char * xmlDesc,
unsigned int flags)

typedef virConnectDomainEventCallback
int virConnectDomainEventCallback (connectPtr conn,
domainPtr dom,
int event,
int detail,
void * opaque)

int virConnectDomainEventDeregister (connectPtr conn,
domainEventCallback cb)

int virConnectDomainEventDeregisterAny (connectPtr conn,
int callbackID)

typedef virConnectDomainEventDiskChangeCallback
void virConnectDomainEventDiskChangeCallback (connectPtr conn,
domainPtr dom)
```
smart developer’s idea: combine all previous ideas!

“We’ll get a fee for the **real doc**, give away the **auto-generated stuff**, and let users **write the rest**.”
smart developer’s idea: combine all previous ideas!

Documentation

CMake has several forms of documentation. There is a book *Mastering CMake* by Ken Martin and Bill Hoffman that can be purchased through [CMake’s e-store](http://cmake.org) or through Amazon.

**Documentation Links:**

- [Mastering CMake](http://cmake.org)
- CMake Tutorial - excerpt from *Mastering CMake*, Chapter 12.
- Reference Documentation:
  - CMake: 2.8.8, 2.8.7, 2.8.6, 2.8.5, 2.8.4, 2.8.3, 2.8.2, 2.8.1, 2.8.0, 2.6.x.
  - Chibi: 2.8.8, 2.8.7, 2.8.6, 2.8.5, 2.8.4, 2.8.3, 2.8.2, 2.8.1, 2.8.0, 2.6.x.
  - Chibi: 2.8.8, 2.8.7, 2.8.6, 2.8.5, 2.8.4, 2.8.3, 2.8.2, 2.8.1, 2.8.0, 2.6.x.
- CMake Wiki Page
- CMake FAQ Page
smart developer’s idea: combine all previous ideas!

CMake 2.8.8 Documentation

Generated by make --help-html.

Master Index CMake 2.8.8

- Name
- Usage
- Description
- Options
- Generators
- Commands
- Properties
- Properties of Global Scope
- Properties on Directories
- Properties on Targets
- Properties on Tests
- Properties on Source Files
- Properties on Cache Entries
- Compatibility Commands
- Standard CMake Modules
- Policies

Inria L. Courtès — Software Documentation—That Thing We Love to Hate
“If you want them to RTFM, make a better FM.”

— unknown author
introduction

soothing the developer’s conscience

meeting the user’s expectations

tools for doc

outro
contents
provide context & motivation

1 Introduction

The C language provides no built-in facilities for performing such common operations as input/output, memory management, string manipulation, and the like. Instead, these facilities are defined in a standard library, which you compile and link with your programs.

The GNU C library, described in this document, defines all of the library functions that are specified by the ISO C standard, as well as additional features specific to POSIX and other derivatives of the Unix operating system, and extensions specific to the GNU system.
provide context & motivation

6.1 Data management

When the application allocates data, whenever possible it should use the `starpu_malloc` function, which will ask CUDA or OpenCL to make the allocation itself and pin the corresponding allocated memory. This is needed to permit asynchronous data transfer, i.e. permit data transfer to overlap with computations. Otherwise, the trace will show that the `DriverCopyAsync` state takes a lot of time, this is because CUDA or OpenCL then reverts to synchronous transfers.
give examples to... help get started

**A simple first program**

Here is a rather simple program to get you started.

```cpp
#include <iostream>
#include <Eigen/Dense>

using Eigen::MatrixXd;

int main()
{
    MatrixXd m(2,2);
    m(0,0) = 3;
    m(1,0) = 2.5;
    m(0,1) = -1;
    m(1,1) = m(1,0) + m(0,1);
    std::cout << m << std::endl;
}
```

We will explain the program after telling you how to compile it.
1.3 Functions as values

Caml is a functional language: functions in the full mathematical sense are supported and can be passed around freely just as any other piece of data. For instance, here is a deriv function that takes any float function as argument and returns an approximation of its derivative function:

```plaintext
# let deriv f dx = function x -> (f(x +. dx) -. f(x)) /. dx;;
val deriv : (float -> float) -> float -> float -> float = <fun>

# let sin' = deriv sin 1e-6;;
val sin' : float -> float = <fun>

# sin' pi;;
- : float = -1.00000000013961143
```
give examples to... illustrate functions

```scheme
substring=? string1 string2 len

This function returns #t if string1 and string2 have a common prefix of size len.

(substring=? "abcdef" "ab9989898" 2)
⇒ #t
(substring=? "abcdef" "ab9989898" 3)
⇒ #f
```

reference doc calls for rigorous wording

**format**

```java
public final String format(Date date)
```

Formats a Date into a date/time string.

**Parameters:**
- `date` - the time value to be formatted into a time string.

**Returns:**
- the formatted time string.
reference doc calls for rigorous wording

format

public final String format(Date date)

Formats a Date into a date/time string.

Parameters:

date - the time value to be formatted into a time string.

Returns:

the formatted time string.
reference doc calls for rigorous wording

— Function: char * **asctime** (const struct tm **brokentime**)

The `asctime` function converts the broken-down time value that `brokentime` points to into a string in a standard format:

"Tue May 21 13:46:22 1991\n"


The return value points to a statically allocated string, which might be overwritten by subsequent calls to `asctime` or `ctime`. (But no other library function overwrites the contents of this string.)
reference doc calls for rigorous wording

— Function: char * asctime (const struct tm *brokentime)

The asctime function converts the broken-down time value that brokentime points to into a string in a standard format:

"Tue May 21 13:45:22 1991\n"


The return value points to a statically allocated string, which might be overwritten by subsequent calls to asctime or ctime. (But no other library function overwrites the contents of this string.)
form
Terms and Definitions

Object
A consistent kind of part of the system, such as a Core, a Cache, a Memory node, etc. The different types detected by hwloc are detailed in the `hwloc_obj_type_t` enumeration.

They are topologically sorted by CPU set into a tree.

CPU set
The set of logical processors (or processing units) logically included in an object (if it makes sense). They are always expressed using physical logical processor numbers (as announced by the OS). They are implemented as the `hwloc_bitmap_t` opaque structure. hwloc CPU sets are just masks, they do not have any relation with an operating system actual binding notion like Linux' cpusets.

Node set
The set of NUMA memory nodes logically included in an object (if it makes sense). They are always expressed using physical node numbers (as announced by the OS). They are implemented with the `hwloc_bitmap_t` opaque structure, as bitmaps.

Bitmap
A possibly-infinite set of bits used for describing sets of objects such as CPUs (CPU sets) or memory nodes (Node sets). They are implemented with the `hwloc_bitmap_t` opaque structure.
void * memset (void *block, int c, size_t size)    [Function]
This function copies the value of c (converted to an unsigned char) into each of the first size bytes of the object beginning at block. It returns the value of block.

wchar_t * wmemset (wchar_t *block, wchar_t wc, size_t size) [Function]
This function copies the value of wc into each of the first size wide characters of the object beginning at block. It returns the value of block.

char * strcpy (char *restrict to, const char *restrict from) [Function]
This copies characters from the string from (up to and including the terminating null character) into the string to. Like memcpy, this function has undefined results if the strings overlap. The return value is the value of to.
don’t be needlessly verbose

**GUINT32\_FROM\_BE()**

```c
#define GUINT32\_FROM\_BE(val) (GUINT32\_TO\_BE (val))
```

Converts a `guint32` value from big-endian to host byte order.

- `val`: a `guint32` value in big-endian byte order
- **Returns**: `val` converted to host byte order

---

**GUINT32\_FROM\_LE()**

```c
#define GUINT32\_FROM\_LE(val) (GUINT32\_TO\_LE (val))
```

Converts a `guint32` value from little-endian to host byte order.

- `val`: a `guint32` value in little-endian byte order
- **Returns**: `val` converted to host byte order

---

**GUINT32\_TO\_BE()**

```c
#define GUINT32\_TO\_BE(val) (GUINT32\_SWAP\_LE\_BE (val))
```

Converts a `guint32` value from host byte order to big-endian.

- `val`: a `guint32` value in host byte order
- **Returns**: `val` converted to big-endian

---

**GUINT32\_TO\_LE()**

```c
#define GUINT32\_TO\_LE(val) (GUINT32\_SWAP\_BE\_LE (val))
```
don’t be needlessly verbose

— Scheme Procedure: bytevector-u8-ref bv index
— Scheme Procedure: bytevector-s8-ref bv index
— Scheme Procedure: bytevector-u16-ref bv index endianness
— Scheme Procedure: bytevector-s16-ref bv index endianness
— Scheme Procedure: bytevector-u32-ref bv index endianness
— Scheme Procedure: bytevector-s32-ref bv index endianness
— Scheme Procedure: bytevector-u64-ref bv index endianness
— Scheme Procedure: bytevector-s64-ref bv index endianness
— C Function: scm_bytevector_u8_ref (bv, index)
— C Function: scm_bytevector_s8_ref (bv, index)
— C Function: scm_bytevector_u16_ref (bv, index, endianness)
— C Function: scm_bytevector_s16_ref (bv, index, endianness)
— C Function: scm_bytevector_u32_ref (bv, index, endianness)
— C Function: scm_bytevector_s32_ref (bv, index, endianness)
— C Function: scm_bytevector_u64_ref (bv, index, endianness)
— C Function: scm_bytevector_s64_ref (bv, index, endianness)

Return the unsigned $n$-bit (signed) integer (where $n$ is 8, 16, 32 or 64) from $bv$ at $index$, decoded according to $endianness$. 
use (hyper)links to concepts, and indexes

5.2 Assignment Functions

These functions assign new values to already initialized complex numbers (see Initializing Complex Numbers). When using any functions with intmax_t or uintmax_t parameters, you must include <stdint.h> or <inttypes.h> before mpc.h, to allow mpc.h to define prototypes for these functions. Similarly, functions with parameters of type complex or long complex are defined only if <complex.h> is included before mpc.h. If you need assignment functions that are not in the current API, you can define them using the MPC_SET_X_Y macro (see Advanced Functions).

— Function: int mpc_set (mpc_t rop, mpc_t op, mpc_rnd_t rnd)

Set the value of rop from op, rounded to the precision of rop with the given rounding mode rnd.
use (hyper)links to concepts, and indexes

<table>
<thead>
<tr>
<th>Concept Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Menu:</td>
</tr>
<tr>
<td>* C extensions:</td>
</tr>
<tr>
<td>* GCC plug-in:</td>
</tr>
<tr>
<td>* heap_allocated attribute:</td>
</tr>
<tr>
<td>* implicit task CPU implementation:</td>
</tr>
<tr>
<td>* output_type_attribute:</td>
</tr>
<tr>
<td>* task:</td>
</tr>
<tr>
<td>* task_attribute:</td>
</tr>
<tr>
<td>* task_implementation:</td>
</tr>
<tr>
<td>* task-based programming model:</td>
</tr>
<tr>
<td>* task_implementation_attribute:</td>
</tr>
</tbody>
</table>
use (hyper)links to concepts, and indexes

### Function Index

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>starpu_asynchronous_copy_disabled</td>
<td>83</td>
</tr>
<tr>
<td>starpu_bcsr_data_register</td>
<td>90</td>
</tr>
<tr>
<td>starpu_bcsr_get_c</td>
<td>95</td>
</tr>
<tr>
<td>STARPU_BCSR_GET_COLIND</td>
<td>95</td>
</tr>
<tr>
<td>starpu_bcsr_get_elmsize</td>
<td>95</td>
</tr>
<tr>
<td>starpu_bcsr_get_firstentry</td>
<td>94</td>
</tr>
<tr>
<td>starpu_bcsr_get_local_colind</td>
<td>95</td>
</tr>
<tr>
<td>starpu_bcsr_get_local_nzval</td>
<td>95</td>
</tr>
<tr>
<td>starpu_bcsr_get_local_rowptr</td>
<td>95</td>
</tr>
<tr>
<td>starpu_bcsr_get_nnz</td>
<td>94</td>
</tr>
<tr>
<td>STARPU_BCSR_GET_NNZ</td>
<td>95</td>
</tr>
<tr>
<td>starpu_bcsr_get_nrow</td>
<td>94</td>
</tr>
<tr>
<td>STARPU_BCSR_GET_NZVAL</td>
<td>95</td>
</tr>
<tr>
<td>starpu_bcsr_get_r</td>
<td>95</td>
</tr>
<tr>
<td>STARPU_BCSR_GET_ROWPTR</td>
<td>95</td>
</tr>
<tr>
<td>starpu_block_data_register</td>
<td>90</td>
</tr>
<tr>
<td>starpu_block_filter_func</td>
<td>98</td>
</tr>
<tr>
<td>starpu_block_filter_func_block</td>
<td>99</td>
</tr>
<tr>
<td>starpu_block_filter_func_vector</td>
<td>99</td>
</tr>
<tr>
<td>STARPU_BLOCK_GET_DEV_HANDLE</td>
<td>94</td>
</tr>
<tr>
<td>starpu_codelet_init</td>
<td>103</td>
</tr>
<tr>
<td>starpu_codelet_pack_args</td>
<td>35</td>
</tr>
<tr>
<td>starpu_codelet_unpack_args</td>
<td>35</td>
</tr>
<tr>
<td>starpu_combined_worker_assign_workerid</td>
<td>130</td>
</tr>
<tr>
<td>starpu_combined_worker_can_execute_task</td>
<td>130</td>
</tr>
<tr>
<td>starpu_combined_worker_get_count</td>
<td>129</td>
</tr>
<tr>
<td>starpu_combined_worker_get_description</td>
<td>130</td>
</tr>
<tr>
<td>starpu_combined_worker_get_id</td>
<td>130</td>
</tr>
<tr>
<td>starpu_combined_worker_get_rank</td>
<td>129</td>
</tr>
<tr>
<td>starpu_combined_worker_get_size</td>
<td>129</td>
</tr>
<tr>
<td>starpu_conf_init</td>
<td>83</td>
</tr>
<tr>
<td>STARPU_CPU</td>
<td>100</td>
</tr>
<tr>
<td>starpu_cpu_worker_get_count</td>
<td>84</td>
</tr>
<tr>
<td>starpu_crc32_be</td>
<td>125</td>
</tr>
<tr>
<td>starpu_crc32_be_n</td>
<td>124</td>
</tr>
<tr>
<td>starpu_crc32_string</td>
<td>125</td>
</tr>
<tr>
<td>starpucsr_data_register</td>
<td>91</td>
</tr>
<tr>
<td>STARPU_CSR_GET_COLIND</td>
<td>96</td>
</tr>
<tr>
<td>starpucsr_get_elmsize</td>
<td>96</td>
</tr>
</tbody>
</table>
6.20 Foreign Function Interface

The more one hacks in Scheme, the more one realizes that there are actually two computational worlds: one which is warm and alive, that land of parentheses, and one cold and dead, the land of C and its ilk.

But yet we as programmers live in both worlds, and Guile itself is half implemented in C. So it is that Guile's living half pays respect to its dead counterpart, via a spectrum of interfaces to C ranging from dynamic loading of Scheme primitives to dynamic binding of stock C library procedures.

- Foreign Libraries: Dynamically linking to libraries.
- Foreign Functions: Simple calls to C procedures.
- C Extensions: Extending Guile in C with loadable modules.
- Modules and Extensions: Loading C extensions into modules.
- Foreign Pointers: Accessing global variables.
- Dynamic EFL: Calling arbitrary C functions.

6.20 Foreign Function Interface

The more one hacks in Scheme, the more one realizes that there are actually two computational worlds: one which is warm and alive, that land of parentheses, and one cold and dead, the land of C and its ilk.

But yet we as programmers live in both worlds, and Guile itself is half implemented in C. So it is that Guile’s living half pays respect to its dead counterpart, via a spectrum of interfaces to C ranging from dynamic loading of Scheme primitives to dynamic binding of stock C libraries.

* Menu:
  * Foreign Libraries: Dynamically linking to libraries.
  * Foreign Functions: Simple calls to C procedures.
  * C Extensions: Extending Guile in C with loadable modules.
  * Modules and Extensions: Loading C extensions into modules.
  * Foreign Pointers: Accessing global variables.
  * Dynamic FFI: Calling arbitrary C functions.

Click `guile-2.x` on a completion to select it. In this buffer, type RET to select the completion near point.

Possible completions are:
- pointer->bytevector
- pointer->procedure
- pointer->string
- pointer->address
- pointer->mem
- pointer?
6.20 Foreign Function Interface

The more one hacks in Scheme, the more one realizes that there are actually two computational worlds: one which is warm and alive, that land of parentheses, and one cold and dead, the land of C and its ilk.

But yet we as programmers live in both worlds, and Guile itself is half implemented in C. So it is that Guile's living half pays respect to its dead counterpart, via a spectrum of interfaces to C ranging from dynamic loading of Scheme primitives to dynamic binding of stock C library procedures.
6.20 Foreign Function Interface

The more one hacks in Scheme, the more one realizes that there are actually two computational worlds: one which is warm and alive, that of parentheses, and one cold and dead, the land of C and its ilk. Live in both worlds, and Guile itself is implemented in C. So it is that Guile's living half pays respect to interfaces to C ranging from dynamic binding of stock C library procedures.

But yet we as programmers live in both worlds, and Guile itself is implemented in C. So it is that Guile's living half pays respect to interfaces to C ranging from dynamic binding of stock C library procedures.
introduction

soothing the developer’s conscience

meeting the user’s expectations

tools for doc

outro
The key to adding options is the code

```python
self.addCommandOption ('welcome_option', 'w', 'welcome',
    help = _("Display the welcome message."))

self.addCommandOption ('lang_option', 'l', 'lang',
    argument = 'language',
    default = 'english',
    help = _("The language to print welcome")
    "Valid values: english, maori"))
```

`addCommandOption` is a function that adds entries that to your application's list of valid options. The function takes the following parameters:

`(self, name, shortOption, longOption, acceptsArgument, default, argumentName, help, category, action, argument)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>self</code></td>
<td><code>self</code></td>
</tr>
<tr>
<td><code>name</code></td>
<td>The key name that will be available in the <code>self.OPTION</code> dictionary when the application is executing.</td>
</tr>
<tr>
<td><code>shortOption</code></td>
<td>The single letter to assign to this option</td>
</tr>
<tr>
<td><code>longOption</code></td>
<td>The long option name. The is prepended with <code>--</code> on the command line.</td>
</tr>
</tbody>
</table>
are you serious?!

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>self</td>
<td>self</td>
</tr>
<tr>
<td>name</td>
<td>The key name that will be available in the self.OPTION dictionary when the application is executing.</td>
</tr>
<tr>
<td>shortOption</td>
<td>The single letter to assign to this option</td>
</tr>
<tr>
<td>longOption</td>
<td>The long option name. The is prepended with -- on the command line.</td>
</tr>
</tbody>
</table>
mixed form/content concerns
no or distinct versioning
ugly
hard to collaborate
hardly consistent typographical conventions
paper-only
SSH(1)

NAME

ssh - OpenSSH SSH client (remote login program)

SYNOPSIS


DESCRIPTION

ssh (SSH client) is a program for logging into a remote machine and for executing commands on a remote machine. It is intended to replace rlogin and rsh, and provide secure encrypted communications between two untrusted hosts over an insecure network. X11 connections and arbitrary TCP ports can also be forwarded over the secure channel.

ssh connects and logs into the specified hostname (with optional user name). The user must prove his/her identity to the remote machine using one of several methods depending on the protocol version used (see below).

If command is specified, it is executed on the remote host instead of a login shell.

The options are as follows:

-1 Forces ssh to try protocol version 1 only.

-2 Forces ssh to try protocol version 2 only.
**SSH(1)**

**NAME**

Ssh (SSH client) — remote login program

**SYNOPSIS**


**DESCRIPTION**

**SSH** (SSH client) is a program for logging in to another host and executing commands on a remote machine. It is intended to replace rlogin, telnet, and rsh by providing encrypted communications between two untrusted hosts over an insecure network. Any standard TCP ports can also be forwarded over the secure channel.

ssh connects and logs into the specified hostname (with optional user name). The user must prove his/her identity to the remote machine using one of several methods, depending on the protocol version used (see below).

If command is specified, ssh invokes the command on the remote host instead of a login shell.

The options are as follows:

- `-1` Forces ssh to try protocol version 1 only.
- `-2` Forces ssh to try protocol version 2 only.
**Unix man pages**

**NAME**

ssh (Remote login program)

**SYNOPSIS**

```bash
```

**DESCRIPTION**

ssh (SSH client) is a program for logging into and executing commands on a remote machine. It is intended to replace rlogin, rsh, and rcp and provide secure remote communications between two untrusted hosts over an insecure network. Regular TCP ports can also be forwarded over the secure channel.

ssh connects and logs into the specified hostname (with optional user name). The user must prove his/her identity to the remote machine using one of several methods depending on the protocol version used (see below).

If command is specified, ssh invokes the command instead of a login shell.

The options are as follows:

- `-1` Forces ssh to try protocol version 1 only.
- `-2` Forces ssh to try protocol version 2 only.

**ssh(1): 800 lines**

**OpenSSL: 1,170 pages in Sect. 3**
```c
#ifndef HWLOC_BITMAP_H
#define HWLOC_BITMAP_H

/** \defgroup hwlocality_bitmap The bitmap API */
* 
* The ::hwloc_bitmap_t type represents a set of objects.
* @{
* 
*/

/** \brief Free bitmap \p bitmap. */
* 
* If \p bitmap is \c NULL, no operation is performed.
* 
void hwloc_bitmap_free(hwloc_bitmap_t bitmap);
```
namespace Eigen {
/** \page TopicAliasing Aliasing 
Statements like <tt>mat = 2 * mat;</tt> or <tt>mat = mat.transpose();</tt> exhibit aliasing.
</b>Table of contents</b>
- \ref TopicAliasingExamples
- \ref TopicAliasingSolution

\section TopicAliasingExamples Examples
Here is a simple example exhibiting aliasing:
<table class="example">
<tr><th>Example</th><th>Output</th></tr>
<tr><td>
\include TopicAliasing_block.cpp
</td>
<td>
\verbatiminclude TopicAliasing_block.out
</td></tr></table>
namespace Eigen {

/** \page TopicAliasing Aliasing
Statements like <tt>mat = 2 * mat;</tt> or <tt>mat = mat.transpose();</tt> exhibit aliasing.

<b>Table of contents</b>
- \ref TopicAliasingExamples
- \ref TopicAliasingSolution

\section TopicAliasingExamples Examples

Here is a simple example exhibiting aliasing:
<table class="example">
<tr><th>Example</th><th>Output</th></tr>
<tr><td>\include TopicAliasing_block.cpp</td><td>\verbatim#include TopicAliasing_block.out</td></tr>
</table>
namespace Eigen {
/** \page TopicAliasing Aliasing
Statements like <tt>mat = 2 * mat;</tt> or <tt>mat = mat.transpose();</tt> exhibit aliasing.

<b>Table of contents</b>
- \ref TopicAliasingExamples
- \ref TopicAliasingSolution

\section TopicAliasingExamples Examples

Here is a simple example exhibiting aliasing:
\begin{table}
\begin{tabular}{|l|l|}
\hline
Example & Output \\
\hline
\include TopicAliasing_block.cpp & \include TopicAliasing_block.out \\
\hline
\end{tabular}
\end{table}

C++, HTML, and \LaTeX{} all mixed up!

and yet it works!
**

In *Eigen*, aliasing refers to assignment statements in which the same matrix (or array or vector) appears on the left and on the right of the assignment operators. Statements like `mat = 2 * mat;` or `mat = mat.transpose();` exhibit aliasing. The aliasing in the first example is harmless, but the aliasing in the second example leads to unexpected results. This page explains what aliasing is, when it is harmful, and what to do about it.

**Table of contents**

- Examples
- Resolving aliasing issues
- Aliasing and component-wise operations
- Aliasing and matrix multiplication
- Summary

### Examples

Here is a simple example exhibiting aliasing:

```cpp
MatrixXi mat(3,3);
mat << 1, 2, 3, 4, 5, 6, 7, 8, 9;
cout << "Here is the matrix mat:\n" << mat << endl;

// This assignment shows the aliasing problem
mat.bottomRightCorner(2,2) = mat.topLeftCorner(2,2);
cout << "After the assignment, mat = \n" << mat << endl;
```

<table>
<thead>
<tr>
<th>Example</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MatrixXi mat(3,3);</code>&lt;br&gt;mat &lt;&lt; 1, 2, 3, 4, 5, 6, 7, 8, 9;&lt;br&gt;cout &lt;&lt; &quot;Here is the matrix mat:\n&quot; &lt;&lt; mat &lt;&lt; endl;</td>
<td>Here is the matrix mat:&lt;br&gt;1 2 3&lt;br&gt;4 5 6&lt;br&gt;7 8 9&lt;br&gt;After the assignment, mat =&lt;br&gt;1 2 3&lt;br&gt;4 1 2&lt;br&gt;7 4 1</td>
</tr>
</tbody>
</table>
val code : char -> int
  Return the ASCII code of the argument.

val chr : int -> char
  Return the character with the given ASCII code. Raise Invalid_argument "Char.chr" if the argument is outside the range 0–255.

val escaped : char -> string
  Return a string representing the given character, with special characters escaped following the lexical conventions of Objective Caml.

val lowercase : char -> char
  Convert the given character to its equivalent lowercase character.

val uppercase : char -> char
  Convert the given character to its equivalent uppercase character.
\texttt{\LaTeX{}}, a false good idea

\url{http://www.latex-project.org/}

\begin{verbatim}
val code : \texttt{char} -> \texttt{int}
  Returns the ASCII code of the given character.

val chr : \texttt{int} -> \texttt{char}
  Return the character with the given ASCII code. Raise \texttt{Invalid_argument} "Char.chr" if the argument is outside the range 0–255.

val escaped : \texttt{char} -> \texttt{string}
  Return a string representing the given character, with special characters escaped following the lexical conventions of Objective Caml.

val lowercase : \texttt{char} -> \texttt{char}
  Convert the given character to its equivalent lowercase character.

val uppercase : \texttt{char} -> \texttt{char}
  Convert the given character to its equivalent uppercase character.
\end{verbatim}

* HeVeA & co., sure, but heavy use of \texttt{latexonly}, \texttt{ifhtml}, etc.
\LaTeX, a false good idea

http://www.latex-project.org/

```latex
val code : char

Returns the code of the current character.

val chr : int -> char

Returns the character with the given ASCII code. Raise \texttt{Invalid_argument} "Char.chr" if the argument is outside the range 0–255.

val escaped : char -> string

Returns a string representing the given character, with special characters escaped following the conventions of the standard library.

val lowercase : char -> char

Converts the given character to its equivalent lowercase character.

val uppercase : char -> char

Converts the given character to its equivalent uppercase character.

* HeVeA & co., sure, but heavy use of latexonly, ifhtml, etc.
```
\begin{ocamlcode}
  type int
\end{ocamlcode}
\index{int@\texttt{int}}
\begin{ocamlcdescription}
  The type of integer numbers.
\end{ocamlcdescription}

\begin{ocamlcode}
exception End_of_file
\end{ocamlcode}
\index{Endoffile@\texttt{End_of_file}}
\begin{ocamlcdescription}
  Exception raised by input functions to signal that the end of file has been reached.
\end{ocamlcdescription}
\begin{ocamldoccode}
  type int
\end{ocamldoccode}
\index{int@	exttt{int}}
\begin{ocamldocdescription}
  The type of integer numbers.
\end{ocamldocdescription}

\begin{ocamldoccode}
  exception End_of_file
\end{ocamldoccode}
\index{Endoffile@	exttt{End_of_file}}
\begin{ocamldocdescription}
  Exception raised by input functions to signal that the end of file has been reached.
\end{ocamldocdescription}
\begin{ocamlcode}
type int
\end{ocamlcode}
\index{int@\verb|int|}
\begin{ocamlcode}
exception End_of_file
\end{ocamlcode}
\index{Endoffile@\verb|End_of_file|}
\begin{ocamlcode}
exception End_of_file
\end{ocamlcode}
\begin{ocamlcode}
exception End_of_file
\end{ocamlcode}
\index{Endoffile@\verb|End_of_file|}
\begin{ocamlcode}
Exception raised by input functions to signal that the end of file has been reached.
\end{ocamlcode}
5.1.1. The Nix expression

Example 5.1. Nix expression for GNU Hello

```nix
{ stdenv, fetchurl, perl }: ①

stdenv.mkDerivation { ②
  name = "hello-2.1.1"; ③
  builder = ./builder.sh; ④
  src = fetchurl { ⑤
    md5 = "70c9c0f39e0c07f762c24f2af223978ad";
  };
  inherit perl; ⑥
}
```

Example 5.1, “Nix expression for GNU Hello (default.nix)” shows a Nix expression for GNU Hello. It's customary to place each package in a separate directory and call the single Nix expression in that directory default.nix. The file has the following elements (referenced from the figure by number):

① This states that the expression is a function that expects to be called with three arguments: stdenv, fetchurl, and perl. They are needed to build Hello, but we don't know how to build them here; that's
5.1 A simple Nix expression

This section shows how to add and test the GNU Hello package to the Nix Packages collection. Hello is a program that prints out the text “Hello, world!”.

To add a package to the Nix Packages collection, you generally need to do three things:

1. Write a Nix expression for the package. This is a file that describes all the inputs involved in building the package, such as dependencies, sources, and so on.
2. Write a builder. This is a shell script that actually builds the package from the inputs.
3. Add the package to the file pkgs/top-level/all-packages.nix. The Nix expression written in the first step is a function; it requires other packages in order to build it. In this step you put it all together, i.e., you call the function with the right arguments to build the actual package.

5.1.1 The Nix expression

Example 5.1 Nix expression for GNU Hello (default.nix)

```nix
{ stdenv, fetchurl, perl }:

stdenv.mkDerivation {
  name = "hello-2.1.1";
  builder = ./builder.sh;
  src = fetchurl {
    md5 = "70dcf9cf9ac07f762c24f2d2290794d";
  };
  inherit perl;
}
```

5.1 A simple Nix expression

This section shows how to add and test the GNU Hello package to the Nix Packages collection. Hello is a program that prints out the text “Hello, world!”.

To add a package to the Nix Packages collection, you generally need to do three things:

1. Write a Nix expression for the package. This is a file that describes all the inputs involved in building the package, such as dependencies, sources, and so on.
2. Write a builder. This is a shell script that actually builds the package from the inputs.
3. Add the package to the file `pkgs/top-level/all-packages.nix`. The Nix expression written in the first step is a function; it requires other packages in order to build it. To direct Nix to use the three together, i.e., you call the function with medium-independent!
DocBook markup

```xml
<book xmlns="http://docbook.org/ns/docbook"
      xmlns:xi="http://www.w3.org/2001/XInclude">
  <info>
    <title>User’s Guide</title>
    <author>
      <personname>
        <firstname>Bob</firstname>
        <surname>Smith</surname>
      </personname>
      <affiliation>
        <orgname>Foobar, Inc.</orgname>
      </affiliation>
      <contrib>Author</contrib>
    </author>
    <date>June 2012</date>
  </info>
</book>
```
<section><title>The Nix expression</title>

<example xml:id='ex'><title>Nix expression for GNU Hello (<textpos>default.nix</filename>)</title>

<programlisting>
{ stdenv, fetchurl, perl }: <co xml:id='ex-co1' />
stdenv.mkDerivation { <co xml:id='ex-co2' />

...  
}
</programlisting>

</example>

<para><xref linkend='ex' /> shows an expression for GNU Hello. It’s actually already in the Packages collection in <filename>hello/ex-1/default.nix</filename>.
<section><title>The Nix expression</title>

<example xml:id='ex'><title>Nix expression for GNU Hello (<filename>hello/ex-1/default.nix</filename>)</title><programlisting>
{ stdenv, fetchurl, perl }: <co xml:id='ex-co1' />
stdenv.mkDerivation { <co xml:id='ex-co2' />
  ...
} </programlisting>
</example>

<para><xref linkend='ex' /> shows an expression for GNU Hello. It’s actually already in the Packages collection in <filename>hello/ex-1/default.nix</filename>.

high-level markup!

focus on content!
DocBook tool chain

$ xsltproc --param html.stylesheet 'style.css'
   --param callout.graphics.extension '.gif'
   --nonet --xinclude
   --output manual.html
/path/to/docbook.xsl  manual.xml
$ xsltproc --param html.stylesheet 'style.css'
  --param callout.graphics.extension '.gif'
  --nonet --xinclude
  --output manual.html
/path/to/docbook.xsl manual.xml

$ dblatex manual.xml
6.20 Foreign Function Interface

The more one hacks in Scheme, the more one realizes that there are actually two computational worlds: one which is warm and alive, that land of parentheses, and one cold and dead, the land of C and its ilk.

But yet we as programmers live in both worlds, and Guile itself is half implemented in C. So it is that Guile’s living half pays respect to its dead counterpart, via a spectrum of interfaces to C ranging from dynamic loading of Scheme primitives to dynamic binding of stock C library procedures.

* Menu:
  * Foreign Libraries
  * Foreign Functions
  * C Extensions
  * Modules and Extensions
  * Foreign Pointers
  * Dynamic FFI


In this buffer, type RET to select the completion near point.

Possible completions are:
  pointer->bytevector  pointer->procedure
  pointer->string      pointer->address
  pointer?            pointer->scan
The more one hacks in Scheme, the more one realizes that there are actually two computational worlds: one which is warm and alive, that land of parentheses, and one cold and dead, the land of C and its ilk.

* Foreign Libraries:: Dynamic linking.
* Foreign Functions:: Simple calls to C procedures.
* C Extensions:: Extending Guile in C.
The more one hacks in Scheme, the more one realizes that there are actually two computational worlds: one which is warm and alive, that land of parentheses, and one cold and dead, the land of C and its ilk.

* Foreign Libraries:: Dynamic linking.
* Foreign Functions:: Simple calls to C procedures.
* C Extensions:: Extending Guile in C.

...
@deffn {Scheme Procedure} bytevector-length @var{bv}
@deffnx {C Function} scm_bytevector_length (@var{bv})
Return the length in bytes of bytevector @var{bv}.
@end deffn

@deftypefun size_t scm_c_bytevector_length (SCM @var{bv})
Likewise, return the length in bytes of bytevector @var{bv}.
@end deftypefun
@deftp {Data Type} {struct starpu_data_filter}
The filter structure describes a data partitioning operation, to be given to the @code{starpu_data_partition} function, see @ref{starpu_data_partition} for an example. The fields are:

@table @asis
@item @code{unsigned nchildren}
Number of parts to partition the data into.

@item @code{void *filter_arg_ptr}
Additional pointer parameter for the filter function, such as the sizes of the different parts.
@end table
@end deftp
The filter structure describes a data partitioning operation, to be given to the `starpu_data_partition` function, see `starpu_data_partition` for an example.

The fields are:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>unsigned nchildren</code></td>
<td>Number of parts to partition the data into.</td>
</tr>
<tr>
<td><code>void *filter_arg_ptr</code></td>
<td>Additional pointer parameter for the filter function, such as the sizes of the different parts.</td>
</tr>
</tbody>
</table>

@end deftp

@end table

@end deftp
GNU Texinfo tool chain

$ makeinfo manual.texi
$ makeinfo --plaintext manual.texi
GNU Texinfo tool chain

$ makeinfo manual.texi
$ makeinfo --plaintext manual.texi

$ makeinfo --html --css-ref=style.css manual.texi
GNU Texinfo tool chain

$ makeinfo manual.texi
$ makeinfo --plaintext manual.texi

$ makeinfo --html --css-ref=style.css manual.texi

$ texi2pdf -I /path/to/texinfo.tex manual.texi
other tools worth a look

- reStructuredText (Python), http://docutils.sf.net/
- Sphinx (Python), http://sphinx.pocoo.org/
- Skribe (Scheme), http://www-sop.inria.fr/mimosa/fp/Skribe/
- Skribilo (Scheme), http://nongnu.org/skribilo/
- ...
introduction

soothing the developer’s conscience

meeting the user’s expectations

tools for doc

outro
summary

- auto-extracted doc is (usually) an insult to users
summary

- auto-extracted doc is (usually) an insult to users
- doc structure must follow human reasoning
summary

- auto-extracted doc is (usually) an insult to users
- doc structure must follow human reasoning
- use tools that separate presentation & content
worthy readings

- “GNOME Documentation Style Guide V1.6”,
  http://developer.gnome.org/gdp-style-guide/
- “GNU Coding Standards”,
  http://gnu.org/prep/standards/standards.html