L4env-core Interface

Bernhard Kauer

May 16, 2006

Contents

1 Introduction

This document lists the public interface of a subset of L4env called L4env-core. The goal is to define the functionality which will be available independently of the underlying kernel or user-level implementation.

The functions in “unsupported functions” sections are mostly unimplemented, internal or debugging functions. They are “unsupported” this means they can be removed or changed without notice and should therefore not be used. Functions in other sections will be supported in future L4env versions and can be used in applications. Please note that the API can change with new kernel and L4env implementations, but the same functionality will be available in the future.

There are many functions not mentioned in this document. Assume them as “unsupported” but ask the current state if you want to use them.

2 Basic Services

2.1 Package names

description Names is a simple name service which provides a mapping of strings to l4_thread_ids and vice versa.

header file *libnames.h*

documentation http://os.inf.tu-dresden.de/l4env/doc/html/names/

name de-/registration

```
int names_register(const char *name);
int names_unregister(const char* name);
```

name query

```
int names_query_name(char *name, l4_threadid_t *id);
int names_waitfor_name(const char* name,
                       l4_threadid_t* id,
                       const int timeout);
```

unsupported functions

```
names_register_thread_weak();
names_unregister_thread();
names_query_id();
names_query_nr();
names_unregister_task();
names_dump();
```

2.2 Package thread

description The L4 Thread Library encapsulates native L4 threads and provides a basic abstraction to the thread related system call interface of the L4 microkernel.

header file *thread.h*

documentation http://os.inf.tu-dresden.de/l4env/doc/html/thread/

create new thread named

```
l4thread_t  l4thread_create_named(void (*func)(void *),
                                  const char*name,
                                  void * data,
                                  l4_uint32_t flag);
```

Supported flags: `L4THREAD_CREATE_ASYNC`

shutdown thread

```
int   l4thread_shutdown(l4thread_t thread);
```

set priority

```
int   l4thread_set_prio(l4thread_t thread, l4_prio_t prio);
```

thread local storage

```
int   l4thread_data_allocate_key(void);
void  l4thread_data_release_key(int key);
int   l4thread_data_set(l4thread_t thread, int key, void * data);
void *l4thread_data_get(l4thread_t thread, int key);
```

thread id handling

```
int            l4thread_equal(l4thread_t t1, l4thread_t t2);
l4thread_t     l4thread_myself(void);
l4_threadid_t  l4thread_l4_id(l4thread_t thread);
l4thread_t     l4thread_id(l4_threadid_t id);
```

unsupported functions

- Thread locking functions
- All functions that deal with the stack (e. g., `l4thread_create_long`, `l4thread_setup`, `l4thread_get_stack`).
- Exit functions.
- Sleep functions. Use l4util function `l4_sleep` instead.
- Funktions that refer implicitely to the `current thread`. The current thread can always be requested by `l4thread_myself`.
- Thread startup synchronization (`l4thread_started`). Use Semaphores instead.

2.3 Package uclibc

description uClibc is the port of the uClibc to L4env. It enables to use basic libc functionality in L4env-based programs.

header file *stdlib.h, stdio.h, alloca.h, string.h, strings.h*

documentation http://os.inf.tu-dresden.de/l4env/doc/html/uclibc/

memory allocation

```
#include <stdlib.h>
void *malloc(size_t size);
void free(void *ptr);
void *calloc(size_t nmemb, size_t size);

#include <alloca.h>
void *alloca(size_t size);
```

string functions

```
#include <stdlib.h>
ato*(..);
strto*(...);

#include <string.h>
mem*(...);
str*(...);

#include <strings.h>
void bzero(void *s, size_t n);
int  bcmp(const void *s1, const void *s2, size_t n);
void bcopy(const void *src, void *dest, size_t n);
```

```
#include <stdio.h>
int snprintf(char *str, size_t size, const char *format, ...);
int sscanf(const char *str, const char *format, ...);

#include <stdarg.h>
int vsnprintf(char *str, size_t size, const char *format, va_list ap);
int vsscanf(const char *str, const char *format, va_list ap);
```

I/O (for debugging)

```
#include <stdio.h>
int printf(const char *format, ...);

#include <stdarg.h>
int vprintf(const char *format, va_list ap);
```

unsupported functions (some functions may be added upon request)

- File I/O
- Networking
- Math
- Time
- Pthreads
- Signals

3 Synchronization

3.1 Package lock

description The l4 lock library is a simple lock implementation.

header file *lock.h*

documentation http://os.inf.tu-dresden.de/l4env/doc/html/lock/

lock operations

```
void l4lock_lock(l4lock_t * lock);
int  l4lock_try_lock(l4lock_t * lock);
void l4lock_unlock(l4lock_t * lock);
l4thread_t l4lock_owner(l4lock_t * lock);
```

3.2 Package semaphore

description The L4 Semaphore library provides a simple, task-local counting semaphore implementation on L4.

header file *semaphore.h*

documentation http://os.inf.tu-dresden.de/l4env/doc/html/semaphore/

configuration

```
int l4semaphore_init(void);
int l4semaphore_set_thread_prio(l4_prio_t prio);
```

semaphore operations

```
void l4semaphore_down(l4semaphore_t * sem);
int  l4semaphore_try_down(l4semaphore_t * sem);
void l4semaphore_up(l4semaphore_t * sem);
```

unsupported functions

```
l4semaphore_down_timed();
```

4 Memory Management

4.1 Package {dm_generic,dm_mem,dm_phys}

description Dataspace Manager are implementing the low level memory handling in L4env.

header file *dm_generic.h, dm_mem.h, dm_phys.h*

documentation http://os.inf.tu-dresden.de/l4env/doc/html/{dm_generic,dm_mem,dm_phys}/

Generic Dataspace functions

```
#include <dm_generic.h>
int l4dm_map(const void * ptr, l4_size_t size, l4_uint32_t flags);
int l4dm_close(const l4dm_dataspace_t * ds);
int l4dm_share(const l4dm_dataspace_t * ds,
               l4_threadid_t client, l4_uint32_t rights);
int l4dm_revoke(const l4dm_dataspace_t * ds,
                l4_threadid_t client, l4_uint32_t rights);
int l4dm_check_rights(const l4dm_dataspace_t * ds, l4_uint32_t rights);
int l4dm_transfer(const l4dm_dataspace_t * ds, l4_threadid_t new_owner);
int l4dm_copy(const l4dm_dataspace_t * ds, l4_uint32_t flags,
              const char * name, l4dm_dataspace_t * copy);
```

Memory Dataspace functions

```
#include <dm_mem.h>
int l4dm_mem_open(l4_threadid_t dsm_id,
                  l4_size_t size,
                  l4_addr_t align,
                  l4_uint32_t flags,
                  const char * name,
                  l4dm_dataspace_t * ds);
int l4dm_mem_size(const l4dm_dataspace_t * ds, l4_size_t * size);
void *l4dm_mem_allocate(l4_size_t size, l4_uint32_t flags);
void l4dm_mem_release(const void * ptr);
int l4dm_mem_ds_phys_addr(const l4dm_dataspace_t * ds,
                          l4_offs_t offset,
```

```
                            l4_size_t size,
                            l4_addr_t * paddr,
                            l4_size_t * psize);
int l4dm_mem_ds_is_contiguous(const l4dm_dataspace_t * ds);
```

Physical Memory Dataspace functions

```
#include<dm_phys.h>
int l4dm_memphys_open(int pool,
                      l4_addr_t addr,
                      l4_size_t size,
                      l4_addr_t align,
                      l4_uint32_t flags,
                      const char * name,
                      l4dm_dataspace_t * ds);
int l4dm_memphys_poolsize(int pool,
                          l4_size_t * size,
                          l4_size_t * free);
l4_threadid_t l4dm_memphys_find_dmphys(void);
```

unsupported functions

```
l4dm_map_pages();
l4dm_map_ds();
l4dm_close_all();
l4dm_copy_long();
l4dm_ds_set_name();
l4dm_ds_get_name();
l4dm_ds_show();
l4dm_ds_list();
l4dm_ds_list_all();
l4dm_mem_resize();
l4dm_mem_info();
l4dm_mem_ds_lock();
l4dm_mem_ds_unlock();
l4dm_mem_lock();
l4dm_mem_unlock();
l4dm_mem_allocate_named();
l4dm_mem_ds_allocate_named();
l4dm_mem_ds_allocate_named_dsm();
l4dm_memphys_copy();
l4dm_memphys_pagesize();
l4dm_memphys_check_pagesize();
l4dm_memphys_show_memmap();
l4dm_memphys_show_pools();
l4dm_memphys_show_pool_areas();
l4dm_memphys_show_pool_free();
l4dm_memphys_show_slabs();
```

These are extended and debug functions (mostly used internally by libs).

4.2 Package l4rm

description The L4 Region Mapper (L4RM) is the part of the L4 environment which manages the virtual address space of a task.

header file *l4rm.h*

documentation http://os.inf.tu-dresden.de/l4env/doc/html/l4rm/

Attach / Detach Dataspaces

```
int l4rm_attach(const l4dm_dataspace_t * ds,
                l4_size_t size,
                l4_offs_t ds_offs,
                l4_uint32_t flags,
                void ** addr);
int l4rm_attach_to_region(const l4dm_dataspace_t * ds,
                          const void * addr,
                          l4_size_t size,
                          l4_offs_t ds_offs,
                          l4_uint32_t flags);
int l4rm_detach(const void * addr);
```

Further restrictions:

- all regions should be size aligned
- `L4RM_MAP` is deprecated, i. e., pages get mapped lazily

Virtual Memory Management

```
void l4rm_set_unkown_pagefault_callback(l4rm_callback_fn_t callback);
```

All other modules and functions are either unsupported or they are only to be used by the l4env core.

5 Device Drivers

5.1 Package generic_io

description The generic_io interface defines the API to device specific resources and is complemented by omega0. The l4io server implements all functionality for this API.

header file *generic_io.h*

documentation http://os.inf.tu-dresden.de/l4env/doc/html/generic_io/

Initialization and memory handling

```
int l4io_init(l4io_info_t **io_info_addr, l4io_drv_t drv_type);
l4_addr_t l4io_request_mem_region(l4_addr_t start,
                                  l4_size_t len,
                                  l4_umword_t *offset);
int l4io_release_mem_region(l4_addr_t start, l4_size_t len);
int l4io_request_region(l4_addr_t start, l4_size_t len);
int l4io_release_region(l4_addr_t start, l4_size_t len);
```

PCI bus handling

```
int l4io_pci_find_device(unsigned short vendor,
                         unsigned short device,
                         l4io_pdev_t start,
                         l4io_pci_dev_t * pci_dev);
int l4io_pci_enable(l4io_pdev_t pdev);
int l4io_pci_disable(l4io_pdev_t pdev);
void l4io_pci_set_master(l4io_pdev_t pdev);
int l4io_pci_set_pm(l4io_pdev_t pdev, int state);
```

PCI config space access

```
int l4io_pci_readl_cfg(l4io_pdev_t pdev, int pos, l4_uint32_t * val);
int l4io_pci_writel_cfg(l4io_pdev_t pdev, int pos, l4_uint32_t val);
```

unsupported functions

```
l4io_search_mem_region();
l4io_request_dma();
l4io_release_dma();
l4io_pci_find_slot();
l4io_pci_find_class();
int l4io_pci_writeb_cfg(l4io_pdev_t pdev, int pos, l4_uint8_t val);
int l4io_pci_writew_cfg(l4io_pdev_t pdev, int pos, l4_uint16_t val);
int l4io_pci_readb_cfg(l4io_pdev_t pdev, int pos, l4_uint8_t * val);
int l4io_pci_readw_cfg(l4io_pdev_t pdev, int pos, l4_uint16_t * val);
```

5.2 Package omega0

description The omega0 interface is provided by the l4io server. We do not support extra omega0 servers.

header file *client.h*

documentation http://os.inf.tu-dresden.de/l4env/doc/html/omega0/

IRQ handling

```
int omega0_attach(omega0_irqdesc_t desc);
int omega0_request(int handle, omega0_request_t action);
int omega0_detach(omega0_irqdesc_t desc);
```

unsupported functions

- Alien interrupts as in `omega0_set_alien_handler()`
- Iterators as in `omega0_first()/omega0_next()`
- `omega0_request_timeout()`
- `omega0_pass()`

5.3 Package ore

description ORe (the Oshkosh Resurrection) is a best-effort network multiplexer for L4 applications.

header file *ore.h*

documentation http://os.inf.tu-dresden.de/l4env/doc/html/ore/

functions for normal driver life cycle

```
int  l4ore_open(char *device, unsigned char mac[6], l4ore_config *conf);
int  l4ore_send(int handle, char *buf, l4_size_t size);
int  l4ore_recv_blocking(int handle,
                         char **buf,
                         l4_size_t *size,
                         l4_timeout_t timeout);
int  l4ore_recv_nonblocking(int handle, char **buf, unsigned int *size);
void l4ore_close(int handle);
```

configuration

```
void l4ore_set_config(int handle, l4ore_config *conf);
```

unsupported functions

```
uip-ore.h
ore-dsi.h
l4ore_debug();
l4ore_get_send_area();
l4ore_get_recv_area();
l4ore_packet_to_pool();
```