libuiomux

libuiomux

UIOMux is a conflict manager for system resources, including UIO devices.

Download

libuiomux-1.6.2.tar.gz

The libuiomux source archive contains:

This source archive contains:

- libuiomux: the libuiomux shared library
- uiomux: a commandline tool for querying and manipulating UIOMux

Source

The source archive is available via git, at github.com/pedwo/libuiomux:

git clone git://github.com/pedwo/libuiomux.git

Overview

UIOMux multiplexes access to named resources, including devices which are available via UIO.

UIOMux consists of a user-level shared library, libuiomux, that provides system-wide UIO memory management and locking. libuiomux can be used to manage contention across multiple simultaneous processes and threads.

UIOMux allows simultaneous locking of access to multiple resources, with deterministic locking and unlocking order to avoid circular waiting. Processes or threads requiring simultaneous access to more than one resource should lock and unlock them simultaneously via libuiomux.

UIO devices report activity through a read of their file descriptor. UIOMux provides a simplified interface for waiting for a UIO managed resource.

UIOMux provides functions for allocating the UIO memory reserved by the kernel for each device. Allocations are cleared on process exit.

Additionally, UIOMux can be queried for whether or not a resource is available on the currently running system.

Commandline tool

```
Usage: uiomux <command>

uiomux is a tool for querying UIO and managing the UIOMux state.

Reporting:
  query       List available UIO device names that can be managed by UIOMux.
  info        Show memory layout of each UIO device managed by UIOMux.
  meminfo     Show memory allocations of each UIO device managed by UIOMux.

Management:
  reset       Reset the UIOMux system. This initializes the UIOMux shared state,
              including all shared mutexes, and scans UIO memory maps.
  destroy     Destroy the UIOMux system. This frees all resources used by the
              UIOMux shared state. Note that any subsequent program using UIOMux
              will reallocate and initialize this shared state, including this
              tool's 'info' and 'reset' commands.

Utilities:
  alloc <n>   Allocate a specified number of bytes.
```

libuiomux API

Documentation

If doxygen is installed, API documentation will be built in doc/libuiomux. HTML documentation is created by default in doc/libuiomux/html. A PDF version can be created if LaTeX is installed; in this case, run 'make' in doc/libuiomux/latex.

Summary

libuiomux can be used to manage various named resources. Currently the following names are defined:

```
UIOMUX_SH_BEU
UIOMUX_SH_CEU
UIOMUX_SH_JPU
UIOMUX_SH_VEU
UIOMUX_SH_VPU
UIOMUX_SH_ISP
```

These identifiers are bitmasked together as the return value from uiomux_query(), or as arguments to uiomux_lock() and uiomux_unlock().

At any time, an application may retrieve a printable name for a resource:

```
const char * name;

name = uiomux_name(resource);
```

To query which resources are available on the running system, call:

```
uiomux_blockmask_t available_resources;

available_resources = uiomux_query();
```

The return value is a bitmask consisting of some of the above symbols OR'd together. An application may use this function to fall back to a software implementation if a needed hardware resource is not available, or to disable relevant funtionality at runtime.

A process or thread wishing to use the locking facilities of libuiomux should start by calling uiomux_open() to obtain a UIOMux* handle:

```
UIOMux * uiomux;
uiomux = uiomux_open();
```

To request exclusive access to a resource or a set of resources, call uiomux_lock():

```
uiomux_blockmask_t resources;
uiomux_lock(resources);
```

where resources is the name of a resource, or multiple resource names OR'd together. Each call to uiomux_lock() must be paired with a corresponding call to uiomux_unlock():

```
uiomux_unlock(resources);
```

Failure to unlock can lead to system-wide starvation of the locked resource. Note however that all locks obtained via libuiomux will be automatically unlocked on program termination to minimize the potential damage caused by rogue processes.

UIO devices report activity through a read of their file descriptor. A simplified interface is offered for waiting for a UIO managed resource:

```
uiomux_sleep (uiomux, resource);
```

UIOMux also provides a co-operative memory allocation scheme. To allocate memory from a UIO managed resource:

```
uiomux_malloc (uiomux, resource, size, alignment);
```

To free allocated memory from a UIO managed resource:

```
uiomux_free (uiomux, resource, address, size);
```

Note that any outstanding allocations are removed on process exit, and any invalid allocations are cleared on uiomux_open().

Finally, each process or thread that opened a UIOMux* handle should close it by calling uiomux_close(). This will remove associated memory maps, unlock locked resources and mark used memory for deallocation:

```
uiomux_close(uiomux);
```

License