

Fusion

High level IPC
July 2003

by Denis Oliver Kropp <dok@directfb.org>

Table of Contents

1	Introduction	1
1.1	What is Fusion?	1
1.2	Master/Slave vs. Client/Server	1
2	Design	2
2.1	Shared memory	2
2.2	Arena	2
2.3	Skirmish	2
2.4	Reference counter	2
2.5	Reactor	2
2.6	Object	2
2.7	Property	3
2.8	Messages	3
2.9	Call	3
3	Implementation	4
3.1	Kernel space	4
3.1.1	Skirmish	4
3.1.2	Reference counter	4
3.1.3	Property	4
3.2	Kernel/user space	4
3.2.1	Messages	4
3.2.2	Call	4
3.2.3	Reactor	4
3.3	User space	4
3.3.1	POSIX shared memory	4
3.3.2	Arena	4
3.3.3	Object	4

1 Introduction

1.1 What is Fusion?

Fusion is a high level IPC API providing mechanisms for master/slave environments.

1.2 Master/Slave vs. Client/Server

In client/server environments clients delegate every operation to the server which has sole access to shared resources. Such environments suffer from the overhead caused by

- Command encoding and decoding
- Transport of command packets
- Transport of data referenced by commands
- Context switches between client and server processes
- Strict separation of client and server implementation

The master/slave approach eliminates this overhead by virtually merging the processes at the lower level of their implementation. This way of making multiple processes look like multiple threads gave Fusion its name and includes

- Shared memory for data normally held solely by the server
- High level messaging for virtually local event handlers and callbacks
- Advanced locking techniques for synchronization and access control
- Framework for distributed allocation and cleanup of resources

2 Design

This chapter explains some fundamental concepts of Fusion.

2.1 Shared memory

Shared memory provides direct access for all processes to many kinds of shared information, including structures, lists and data. It's also used by the user space part of Fusion itself.

Fusion's shared memory has the following special features

- Shared memory is mapped to the same virtual address in all processes to enable pointers within shared memory pointing to other locations in shared memory
- Allocation and deallocation of shared memory blocks is provided by an API that is equivalent to the libc malloc(), realloc(), free() etc.

2.2 Arena

The first process in a master/slave environment initializes the session while additional processes simply join. The Fusion Arena represents such an environment and defines the role of each entering process. Each Arena is identified by a name which is specified when the Arena is entered. If the Arena doesn't exist yet, the process becomes the master and initializes the rest of the environment. Otherwise the process becomes a slave and joins the environment running the local setup only.

The Fusion Arena also provides "fields" which are basically pairs of a string and a shared memory pointer. These fields are used to register entry points to the shared memory for each part of the environment during its initialization. They are looked up by their name when the environment is being joined by another process.

The exit mechanism is similar to the enter mechanism. If the exiting process is the last one, it does a complete shutdown, otherwise it simply leaves. The master may reject to leave the environment before all slaves did. In this case the exit fails and the master can notify the user, wait for all slaves or kill them.

2.3 Skirmish

2.4 Reference counter

2.5 Reactor

2.6 Object

2.7 Property

2.8 Messages

2.9 Call

3 Implementation

3.1 Kernel space

3.1.1 Skirmish

3.1.2 Reference counter

3.1.3 Property

3.2 Kernel/user space

3.2.1 Messages

3.2.2 Call

3.2.3 Reactor

3.3 User space

3.3.1 POSIX shared memory

3.3.2 Arena

3.3.3 Object