## NAME

**driver** - structure describing a device driver

## SYNOPSIS

```
#include <sys/param.h>
#include <sys/kernel.h>
#include <sys/bus.h>
#include <sys/module.h>

static int foo_probe(device_t);
static int foo_attach(device_t);
static int foo_detach(device_t);
static int foo_frob(device_t, int, int);
static int foo_twiddle(device_t, char *);

static device_method_t foo_methods[] = {
        /* Methods from the device interface */
        DEVMETHOD(device_probe,         foo_probe),
        DEVMETHOD(device_attach,        foo_attach),
        DEVMETHOD(device_detach,        foo_detach),

        /* Methods from the bogo interface */
        DEVMETHOD(bogo_frob,            foo_frob),
        DEVMETHOD(bogo_twiddle,         foo_twiddle),

        /* Terminate method list */
        DEVMETHOD_END
};

static driver_t foo_driver = {
        "foo",
        foo_methods,
        sizeof(struct foo_softc)
};

DRIVER_MODULE(foo, bogo, foo_driver, NULL, NULL);
```

## DESCRIPTION

Each driver in the kernel is described by a driver_t structure. The structure contains the name of the device, a pointer to a list of methods, an indication of the kind of device which the driver implements

and the size of the private data which the driver needs to associate with a device instance. Each driver will implement one or more sets of methods (called interfaces). The example driver implements the standard "driver" interface and the fictitious "bogo" interface.

When a driver is registered with the system (by the DRIVER_MODULE macro, see DRIVER_MODULE(9)), it is added to the list of drivers contained in the devclass of its parent bus type. For instance all PCI drivers would be contained in the devclass named "pci" and all ISA drivers would be in the devclass named "isa". The reason the drivers are not held in the device object of the parent bus is to handle multiple instances of a given type of bus. The DRIVER_MODULE macro will also create the devclass with the name of the driver and can optionally call extra initialisation code in the driver by specifying an extra module event handler and argument as the last two arguments.

## SEE ALSO

devclass(9), device(9), DEVICE_ATTACH(9), DEVICE_DETACH(9), DEVICE_IDENTIFY(9), DEVICE_PROBE(9), DEVICE_SHUTDOWN(9), DRIVER_MODULE(9)

## HISTORY

The **driver** framework first appeared in FreeBSD 2.2.7.

## AUTHORS

This manual page was written by Doug Rabson.