NAME

openprom – PROM monitor configuration interface

SYNOPSIS

#include <sys/fcntl.h>
#include <sys/types.h>
#include <sys/openpromio.h>

open("/dev/openprom", *mode***);**

DESCRIPTION

The internal encoding of the configuration information stored in EEPROM or NVRAM varies from model to model, and on some systems the encoding is "hidden" by the firmware. The **openprom** driver provides a consistent interface that allows a user or program to inspect and modify that configuration, using **ioctl**(2) requests. These requests are defined in **<sys/openpromio.h>**:

```
struct openpromio {
        u_int   oprom_size;                /* real size of following array */
        char    oprom_array[1];            /* For property names and values */
                                           /* NB: Adjacent, Null terminated */
};
#define OPROMMAXPARAM    32768             /* max size of array */
```

For all **ioctl**(2) requests, the third parameter is a pointer to a **'struct openpromio'**. All property names and values are null-terminated strings; the value of a numeric option is its ASCII representation.

IOCTLS

OPROMGETOPT This ioctl takes the null-terminated name of a property in the *oprom_array* and returns its null-terminated value (overlaying its name). *oprom_size* should be set to the size of *oprom_array*; on return it will contain the size of the returned value. If the named property does not exist, or if there is not enough space to hold its value, then *oprom_size* will be set to zero. See BUGS below.

OPROMSETOPT This ioctl takes two adjacent strings in *oprom_array*; the null-terminated property name followed by the null-terminated value.

OPROMSETOPT2 This ioctl is similar to **OPROMSETOPT**, except that it uses the difference between the actual user array size and the length of the property name plus its null terminator.

OPROMNXTOPT This ioctl is used to retrieve properties sequentially. The null-terminated name of a property is placed into *oprom_array* and on return it is replaced with the null-terminated name of the next property in the sequence, with *oprom_size* set to its length. A null string on input means return the name of the first property; an *oprom_size* of zero on output means there are no more properties.

OPROMNXT, **OPROMCHILD**, **OPROMGETPROP**, and **OPROMNXTPROP**
These ioctls provide an interface to the raw *config_ops* operations in the PROM monitor. One can use them to traverse the system device tree; see **prtconf**(1M).

OPROMGETVERSION This ioctl returns an arbitrary and platform-dependent NULL-terminated string in *oprom_array*, representing the underlying version of the firmware.

ERRORS

EAGAIN There are too many opens of the **/dev/openprom** device.

EFAULT A bad address has been passed to an **ioctl**(2) routine.

EINVAL The size value was invalid, or (for **OPROMSETOPT**) the property does not exist, or and invalid ioctl is being issued.

ENOMEM The kernel could not allocate space to copy the user's structure.

EPERM Attempts have been made to write to a read-only entity, or read from a write only entity.

ENXIO Attempting to open a non-existent device.

EXAMPLES

The following example shows how the *oprom_array* is reused for the copied out data.

```
/*
 * This program opens the openprom device and prints the platform
 * name (root node name property) and the prom version.
 *
 * NOTE: /dev/openprom is readable only by user 'root' or group 'sys'.
 */

#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/openpromio.h>

#define BUFSZ           4096      /* Arbitrary buffer size */
#define MAXNAMESZ       32        /* Maximum property name size */
#define MAXVALSZ        (BUFSZ - MAXNAMESZ - sizeof (u_int))
typedef union {
        char            buf[BUFSZ];
        struct openpromio opp;
} Oppbuf;

static char *promdev = "/dev/openprom";

/*
```

```
 * Get the peer node of the given node.  The root node is the peer of zero.
 * After changing nodes, property lookups apply to that node.  The driver
 * 'remembers' what node you are in.
 */
static int
peer(int nodeid, int fd)
{
        Oppbuf oppbuf;
        struct openpromio *opp = &(oppbuf.opp);
        int *ip = (int *)(opp->oprom_array);

        (void) memset(oppbuf.buf, 0, BUFSZ);
        opp->oprom_size = MAXVALSZ;
        *ip = nodeid;
        if (ioctl(fd, OPROMNEXT, opp) < 0) {
                perror("OPROMNEXT");
                exit(1);
        }
        return (*(int *)opp->oprom_array);
}

int
main(void)
{
        Oppbuf oppbuf;
        struct openpromio *opp = &(oppbuf.opp);
        int fd;

        if ((fd = open(promdev, O_RDONLY)) < 0)  {
                fprintf(stderr, "Cannot open openprom device0);
                exit(1);
        }

        /*
         * Get and print the value of the root node 'name' property
         */

        (void) peer(0, fd);                /* Navigate to the root node */

        (void) memset(oppbuf.buf, 0, BUFSZ);
        opp->oprom_size = MAXVALSZ;
        (void) strcpy(opp->oprom_array, "name");
        if (ioctl(fd, OPROMGETPROP, opp) < 0) {
                perror("OPROMGETPROP");
                exit(1);
```

```
        }

        if (opp->oprom_size != 0)
                printf("Platform name <%s>\n", opp->oprom_array);

        /*
         * Get and print the prom version.
         */

        opp->oprom_size = MAXVALSZ;
        if (ioctl(fd, OPROMGETVERSION, opp) < 0) {
                perror("OPROMGETVERSION");
                exit(1);
        }

        printf("Prom version <%s>\n", opp->oprom_array);
        (void) close(fd);
        return (0);
}
```

FILES

/dev/openprom PROM monitor configuration interface

SEE ALSO

eeprom(1M), **monitor**(1M), **prtconf**(1M), **mem**(7D)

BUGS

There should be separate return values for non-existent properties as opposed to not enough space for the value.

An attempt to set a property to an illegal value results in the PROM setting it to some legal value, with no error being returned. An **OPROMGETOPT** should be performed after an **OPROMSETOPT** to verify that the set worked.

The driver should be more consistent in its treatment of errors and edge conditions.