

CS-228 Coursework

Due date: 21st April

February 14, 2008

1 Introduction

Operating systems use kernel modules as a mechanism for implementing optional functionality, including device drivers. Your assignment is to create a kernel module for the FreeBSD operating system (version 6 or later) which implements a `/dev/fib` device from which the Fibonacci sequence can be read.

The Fibonacci sequence is defined inductively as:

$$fib(0) = 1$$

$$fib(1) = 1$$

$$fib(n) = fib(n - 1) + fib(n - 2)$$

Your solution should therefore give the sequence 1, 1, 2, 3, 5, 8, 13...

2 Required Component: 40%

Create a FreeBSD kernel module which exports `/dev/fib`, a device which allows the Fibonacci sequence to be read by a userspace process. This should be built using the BSD Make utility and should be written in C99.

Running 'make' should produce a kernel module named `fib.ko`. This basic functionality is worth 10%.

If using "kldload ./fib.ko" works without errors and loads the driver then another 10% will be awarded, with another 10% if `/dev/fib` appears.

A userspace process should then be able to open the device and read 40 bytes of data. This should read the first 10 Fibonacci numbers as a sequence of 32-bit integers. If this, and all subsequent steps succeed then 40% will be awarded.

Note: Efficiency of generating the Fibonacci sequence will *not* be assessed, however code quality marks may be deducted for any solution with a complexity greater than $O(n)$.

3 Optional Component

The Fibonacci sequence device should allow users to seek to a specific offset in the file. Since the results are returned as 32-bit values, seeking should be done

in units of four bytes. Seeking to offset n and then reading 4 bytes should give Fibonacci number $n/4$. For example, the following snippet of userspace code should result in the value of `n` being set to the 10th Fibonacci number (55):

```
int fib = open("/dev/fib", O_RDWR);
lseek(fib, 40, SEEK_SET);
uint32_t n;
read(fib, &n, 4);
```

10% of the marks are available for a working `ioctl` which allows the sequence seed to be redefined. In terms of the inductive definition on the previous page, this `ioctl` should redefine the value of $fib(1)$. A corresponding `ioctl` should be added to read the current value of the seed.

The `ioctl` command should be defined and used in the following way:

```
#define SET_SEED _IOW('m', 1, int)
int seed = 2;
int error = ioctl(foo, SET_SEED, &seed);
```

After doing this and seeking to the start, the sequence should be 1, 2, 3, 5, 8, and so on.

Representing the Fibonacci sequence with 32-bit integers isn't ideal, since they overflow quickly. A third `ioctl` should be added to allow the user to select between 32-bit and 64-bit integers, and single and double precision floating point values. This is worth an additional 10%.

A further 30% will be awarded for code quality. This includes comments, proper indenting, sensible variable and method names and so on.

4 Useful References

The UNIX manual command, `man`, provides access to the system documentation. Many of the kernel interfaces are documented in section 9 of the manual. Relevant pages include:

`uio(9)`, `DECLARE_MODULE(9)`, `module(9)`, `ioctl(2)`

You may also find the following tutorial useful:

<http://ezine.daemonnews.org/200010/blueprints.html>

The FreeBSD Architecture Handbook includes a short section on writing character devices:

http://www.freebsd.org/doc/en_US.ISO8859-1/books/arch-handbook/

The following paper, presented at BSDCan last year, contains a summary of all of the relevant interfaces:

<http://www.bsdcn.org/2006/papers/freebsd.driver.pdf>