CIS–4020 Lab
Device Drivers

Last Revised: November 25, 2012

1 Introduction

In this lab you will write a simple character device driver for QNX. Normally device drivers interact more or less directly with hardware. However, to make this lab feasible without special equipment the driver you will write here will be for a pure software device. This means that certain aspects of driver construction, such as interrupt handling, will not be covered by this lab. However, you will get experience interfacing the driver to the rest of the operating system and interacting with the driver from other applications.

For purposes of demonstration we will implement a random number generator as a device driver. Normally a random number driver would interact with hardware to use hardware originated events as the source of randomness. The driver you will write for this lab will use a software only pseudo-random number generator.

2 Random Numbers

Methods for generating random numbers abound and much has been written on the subject. For this lab we will use a linear congruential generator (LCG). Such generators are simple and fast but not suitable for high security applications. They can also have other problems if they are not tuned carefully.

LCGs are represented by the formula

$$X_{n+1} = aX_n + c \pmod m$$

This formula allows one to compute the next value in the sequence X_{n+1} from the "current" value X_n. The value X_0 is usually called the *seed* and it should be arbitrary. For purposes of this lab you can use zero. The values (a, c, m) are parameters of the generator and are all constants. Many LCGs use a modulus m that is a power of two since computing the modulus is then just a matter of keeping $\log_2 m$ bits of each result.

The precise properties of the generator depend on the specific parameters (a, c, m) that are used. For our purposes use the parameters

$$\begin{aligned} a &= 1103515245 \\ c &= 12345 \\ m &= 2^{32} \end{aligned}$$

Each time a value X_{n+1} is generated the pseudo-random number output is taken from bits $[30 \ldots 16]$ of X_{n+1}. Notice that despite using a 32 bit internal register, the generator outputs pseudo-random numbers that are only 15 bits in size.

A realistic random number driver would return raw binary data to the applications. However, to make your driver a little easier to test you should return either the character '0' or '1' depending on the least significant bit of the output value (corresponding to bit 16 of X_{n+1}). This means your driver will return printable text that can be displayed directly using a program such as `cat`.

3 Driver

Your driver should be a QNX resource manager. Start with the resource manager example provided in class. You should be able to compile that program as-is and demonstrate its operation. Do this before proceeding. Next you should modify the handling of the io_read message so that it returns the requested number of random values.

Because of the nature of this driver, there is no need to maintain separate state information for different applications that might be using the driver at the same time.

Consult the QNX help documents for more information about resource managers and the functions in the resource manager library.

4 Report

Write a report for this lab following the lab report template provided by your instructor. Include the significant parts of your code, some comments about how it works, and what you observed.