Project 2 Specification
Kernel Module Programming
Kernel Compilation, Kernel Modules, and Scheduling

Final Submission Due: 3/23 at 11:59:59pm

Language Restrictions: C only

Purpose

This project introduces you to the nuts and bolts of kernel compilation, kernel programming, and concurrency and synchronization in the kernel. This project is divided into three parts.

Part 1: Compile a Kernel

(See Project 2A Specification)

Part 2: `remember` Kernel Module

(See Project 2A Specification)

Part 3: The Penguin Printer

(Note: This part of the project does not have anything to do with lp or any pre-existing print drivers or printers.)

Congratulations, you are the proud owner of a new virtual printer called the penguin printer (a.k.a "penguin"). Your task is to implement a printer scheduling algorithm. A printer is defined as a device that accepts print jobs for documents into a queue and processes those documents. A maximum number of jobs will be given. When a printer is loaded, it is initially not running (although it can accept jobs). A printer starts processing jobs with a start command. Each job takes a specific amount of time to process. Finally, a printer must be stopped before it can be unloaded.

Your printer must keep track of the number and type of jobs in a queue. Jobs can come in at any time and can be put on the queue instantaneously. A job must always be accepted if there is room. Jobs can only be processed by the printer one at a time, and it takes the printer 1 second to look inside any spot in the queue (whether it holds a job or not). Once a job is processed, the spot in the queue is cleared and the printer can look for other jobs.

Task Specification

This is a classic exercise in modeling consumers and producers. The producer produces jobs and the consumer is the printer. There are many pieces needed to provide a complete implementation discussed below.

Step 1: Develop a penguin printer /proc module

Develop a `/proc` entry named `/proc/penguin`. In this project, you will be required to support a printer job queue of size 20. You will accept the following 4 jobs:

- 1-page document (internally represented with a 1)
- 2-page document (internally represented with a 2)
- 3-page document (internally represented with a 3)
- 4-page document (internally represented with a 4)

As in a real printer, it takes varying time to process different types of jobs. Your printer must take 2 second to process a 1-page document, 3 seconds to process a 2-page document, 4 seconds to process a 3-page document, and 5 seconds to process a 4-page document. This processing time is ***in addition*** to the 1 second it must take to look in any spot in the queue for an job.

Finally, the printer can break down. It also must accept the following job:

- maintenance (internally represented with a 5)

Maintenance takes a whopping 8 seconds, in addition to the 1 second it must take to look in any spot in the queue for a job.

(Hint – you can "process" for a given amount of seconds by using the ssleep() function.)

You will place jobs on your queue by writing your jobs's number to the /proc/penguin file. For example, the following will put a 2-page document on the order queue:

```
$> echo 2 > /proc/penguin
```

If the queue is full, the printer code should return –ENOMEM and the job should not be placed on the queue.

```
$> echo 2 > /proc/penguin
bash: echo: write error: Cannot allocate memory
```

In addition to accepting jobs 1-5, your printer should accept two additional numbers as commands:

- 0 : start the printer
- -1: stop the printer

When stopping the printer, the printer should wait (`ssleep()`) for 8 seconds so that any maintenance job being processed can finish.

Step 2: Reading from `/proc/penguin`

In addition to accepting jobs and commands, your printer must be able to display status information by reading the device. Specifically, when someone reads from `/proc/penguin`, they should see:

- Printer status: running or not running
- Current spot being looked at in the queue
- Current job being processed

(If the penguin is not running, the last two pieces of information do not need to be displayed.)

For example, take a look at this sample session:

```
$> insmod penguin.ko
$> cat /proc/penguin
Printer is not running.  Total processed is 0.
$> echo 0 > /proc/penguin
$> cat /proc/penguin
Printer is running. Processing nothing at slot 2. Total processed is 0.
$> echo 4 > /proc/penguin
$> cat /proc/penguin
Printer is running. Processing nothing at slot 19.  Total processed is 0.
$> cat /proc/penguin
Printer is running. Processing 4-page document at slot 0. Total processed is 0.
$> cat /proc/penguin
Printer is running. Processing 4-page document at slot 0. Total processed is 0.
$> cat /proc/penguin
Printer is running. Processing nothing at slot 2. Total processed is 1.
$> echo -1 > /proc/penguin
Printer is not running.  Total processed is 1.
$> rmmod penguin
```

The insmod and rmmod commands load and unload the kernel module, respectively. After starting the printer, I placed 4-page document on the queue. However, when immediately reading from /proc/penguin, my printer was not yet looking in the spot in the queue with the 4-page document. Finally, it looks in the right spot and "processes" the 4-page for 5 seconds, so I was able to read from /proc/penguin twice and get the same processing status.

Step 3: Use a kthread

You must use a kthread to allow the printer to run in the background and process jobs in response to a start command. Please look at my example, which starts a kthread on module load. You will need to modify the kthread to process a printing queue and instead start/stop the kthread through commands written to `/proc/penguin`.

Step 4: Extra credit

The top five submissions as measured by the below evaluation procedure will receive +5 points to their project 2 grade. The metric to optimize is: **total jobs processed.**

Halfway Project Submission Procedure
By the halfway point, you should have parts 1 and 2 finished. (Worth 20%)
(See Project 2A Specification)

Full Project Submission Procedure
By the final submission date, part 3 should be finished. (Worth 40%)

. You will need to zip up the following files for submission:

- A folder called Part 3 containing:
    - The penguin.c file **(not the .ko file)**
    - The Makefile for penguin
- The README text file, which should contain:
    - The names of all the members in your group

- A listing of all files/directories in your submission and a brief description of each
- Instructions for compiling your programs (NOTE: use the Makefile provided for you)
- Instructions for running your programs/scripts
- Any challenges you encountered along the way
- Any sources you used to help you write your programs/scripts