

Assignment 3 (Cooperative multitasking)

09 March 2011

due 2 March at noon

In this short assignment, you will experiment with task switching in GeekOS. To retrieve the files, just do an `svn update` again, and the `a3` folder should appear. Build and run the project as it is.

```
$ cd ~/cs643
$ svn update
At revision 148
$ cd a3/build
$ make
```

Task 1

When you run the kernel in VirtualBox, it should print out something like this:

```
8192KB memory detected, 1679 pages in freelist, 1048576 bytes in kernel heap
Initializing IDT...
Initializing keyboard...
Welcome to GeekOS!
Shiny happy people...
```

Now take a look at `a3/src/geekos/main.c`. You should see two functions at the top, called `my_thread_1()` and `my_thread_2()`, and within `main()` you will see two calls to `Start_Kernel_Thread()`.

The two thread functions cooperate to print out the message. Explain what happens during the `Yield()` call in `my_thread_1()`. [Please type your answer into a program comment in `main.c`, just before the `my_thread_1` function.]

Task 2

Now, **just by moving** the `Yield()` call to a different position, force the threads to print out “happy Shiny people...” instead. **do not change or adjust the order of** the `Print()` statements!

Task 3

Now, you will add a `for` loop to each of the thread functions. In `my_thread_1()`, it should loop from 0 to 99 and print out numbers preceded by A, as in:

```
A00 A01 A02 A03 A04 A05 A06 A07 A08 A09 A10 A11 A12 ...
```

To do this, it's helpful to know that the `Print()` function works very much like the standard C `printf()` function: it takes a format string which can contain **percent codes**, like this:

```
Print("A%02d ", i);
```

In this case, the %02d tells it to print a decimal (base 10) integer, in a field of width 2, right-aligned with zeroes. (That's how I get A00 instead of just A0.) Eventually the numbers will wrap around to the next line; that's fine.

Do the same thing for my_thread_2(), only precede the numbers by B. Neither thread should output their numbers until **after** the “happy Shiny people...” message is finished; this may require adding one more Yield() statement somewhere.

Task 4

Now, you will position Yield() statements **inside** the for loops in such a way that my_thread_1() will yield after printing any **odd** number, and my_thread_2() will yield after printing any number divisible by 3. Remember: the C **mod** operator (%) is useful for checking these conditions.

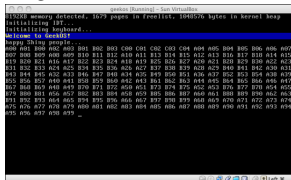
The output should be something like this:

happy Shiny people...

A00 A01 B00 A02 A03 B01 B02 B03 A04 A05 B04 B05 B06 ...etc.

Task 5

Finally, add a new thread function my_thread_3() that prints the numbers 0 through 4 preceded by C. Position calls to Yield() in such a way that the C numbers appear between B03 and A04. At the very least, they should not disrupt the “happy Shiny people...” message.



That's it, commit your work!