ECGR 4101/5101 - Lab 2

Objective: Using standard C libraries on bare-metal ARM

Outcomes:
After this lab, you will
- Install a embedded version of a standard C library
- Examine a minimal implementation of system calls needed to support the C library
- Use GNU Make
- Use the strace utility in Linux to monitor system calls

Using Newlib C library When dealing with embedded software, we often use standard C functions such as: printf, malloc, getchar, strncpy etc. A common way to have them is using Newlib. Newlib is an implementation of the standard C library that is specifically designed to run on a platform with low resources and undefined hardware. The idea of Newlib is to implement the hardware-independent parts of the standard C library and rely on few low-level system calls that must be implemented with the target hardware in mind. Download Newlib C library. Use newlib-2.5.0

```
$ ftp://sourceware.org/pub/newlib/newlib-2.5.0.tar.gz
$ tar xzf newlib-2.5.0.tar.gz
$ cd newlib-2.5.0
$ ./configure --target arm-none-eabi --disable-newlib-supplied-syscalls
$ make // Takes a while!
```

We emulate the RealView platform with an ARM Cortex-A8 processor
Download the following lab2.zip from Canvas into a Lab2 folder. It contains the following files -
mem alloc.c startup.s mem alloc.ld syscalls.c
syscalls.c implements the system calls based on the QEMU emulation of the Versatile Platform Baseboard Cortex-A8 platform. According to the RealView Platform Baseboard for Cortex-A8 User Guide, there is a memory-mapped UART serial port at address 0x10009000. The I/O is done using this UART. The sbrk function is used to increase the memory allocated by the program. If an OS is used, the system calls are implemented by the OS kernel.

```
$ arm-none-eabi-gcc -mcpu=cortex-a8 -I ./newlib-2.5.0/newlib/libc/include
-c -o mem_alloc.o mem_alloc.c
$ arm-none-eabi-as -mcpu=cortex-a8 -o startup.o startup.s
$ arm-none-eabi-gcc -mcpu=cortex-a8 -I ./newlib-2.5.0/newlib/libc/include
-c -o syscalls.o syscalls.c
$ arm-none-eabi-gcc -nostdlib -T mem_alloc.ld mem_alloc.o startup.o
syscalls.o ../newlib-2.5.0/arm-none-eabi/newlib/libc.a /usr/lib/gcc/arm-
none-eabi/4.9.3/libgcc.a -o mem_alloc
```

```
Note: Please use your version of gcc under /usr/lib/gcc/arm-non-eabi/
$ arm-none-eabi-objcopy -O binary mem_alloc mem_alloc.bin
$ ./arm-softmmu/qemu-system-arm -M realview-pb-a8 -nographic -kernel
mem_alloc.bin
```

Each keystroke the program tries to allocate more and more memory until the heap is consumed. To exit QEMU press ctrl-a followed by x.

Do the following:

1. Use the GNU make utility to implement all the compiling and linking steps given above. For more information, http://www.gnu.org/software/make/manual/make.html
2. The implementation of read system call in syscalls.c polls when the receive FIFO is empty. Modify the write system call so that it polls when the transmit FIFO is full.
3. Write a version of mem alloc.c for x86/Linux. Trace the system calls made using the strace Linux utility. Compare the system call trace obtained to those from executing a static binary (compile with -static flag).