The sysfs Virtual File System

- a virtual filesystem available under recent Linux kernels
- provides access to devices and drivers that would otherwise only be accessible within a restricted kernel space

ON-BOARD LEDS

* the following may require root access *

```
BBB $ cd /sys/class/leds
BBB $ ls
    beaglebone:green:usr0   beaglebone:green:usr2
    beaglebone:green:usr1   beaglebone:green:usr3
```

→ these (virtual) directories are LEDS sysfs mappings, each of which corresponds to an on-board LED of the beaglebone ←

```
BBB $ cd beaglebone\:green\:usr3
BBB $ ls
    brightness  device  max_brightness  power  subsystem
    trigger  uevent
```

→ these are virtual files that provide more settings for the USR3 LED ←

You can use the cat/more command to view these files, e.g.)

```
BBB $ cat trigger
    none nand-disk mmc0 [mmc1] timer oneshot
    heartbeat backlight gpio cpu0 ...
```

→ here it can be seen that USR3 is triggered by the eMMC

- please explore some of the other features yourself, and check out the makeLED.cpp program by Dr. Derek Molloy to see how sysfs can be used from within a program

GPIOs

- example to enable GPIO #49, a.k.a. GPIO1_17; 49 = (1 x 32) + 17.

```
BBB $ cd /sys/class/gpio
BBB $ ls
      export gpiochip0 gpiochip32 gpiochip64 gpiochip96
      unexport
BBB $ echo 49 > export
BBB $ ls
      export gpio49 gpiochip0 gpiochip32 gpiochip64
      gpiochip96 unexport
```

new directory emerges

```
BBB$ cd gpio49
BBB $ ls
      active_low direction edge power subsystem
      uevent value
BBB $ cat direction
in
BBB $ echo out > direction
BBB$ cat direction
out
```

gpio49 → now configured as an output

Now we can send values to the output as follows:

```
BBB$ echo 1 > value
```

← turns the output ON

```
BBB$ echo 0 > value
```

← turns the output OFF

We could also remove the gpio from sysfs by issuing the command:

```
BBB$ echo 49 > unexport
```

N.B.: using sysfs on the Beaglebone, the highest frequency with which we can alternate the output is approximately 3 kHz ← not very fast!