**LabVIEW 101 – Weeks 6**

Before class:

1. Download updates from <https://github.com/rizett/LabVIEW-101>

Required equipment:

Computer with LV installed

Topics:

Arrays and reading from file

Running while loops concurrently

Exercises

**Arrays and clusters (briefly):**

Open a blank VI:

* Review difference between Array and Cluster
* How to initialize them
  + Cluster: bundle order depends on order that you put items into the cluster
* What dimensions mean / refer to
  + Arrows at side of array indicator / control / constant refer to index position

**Reading from file**

Open /examples & functions/examples/week5\_read-file.vi

* **Read .lvm file** 
  + Set directory / input file to /examples & functions/examples/example\_lvm\_file.lvm and open the file externally
  + Set the enum select example to Read .lvm file and run
  + Note what happens
  + Note the Read from Meas File settings
    - Set to read .lvm file
  + Now change the file to /examples & functions/examples/example\_lvm\_file\_no-headers.lvm and try to run the program
* **Read .csv file** 
  + Set directory / input file to /examples & functions/examples/example\_csv\_file.csv and open the file externally
  + Set the enum select example to Read .csv file and run
  + Note what happens
* **Read mixed string and num**
  + Set directory / input file to /examples & functions/examples/example\_mixed\_file.csv and open the file externally
  + Set the enum select example to Read mixed string and num and run
  + Note what happens
* **All examples / functions on File I/O palette** 
  + There is also Read Text File function – which just takes a file name and spits out a large chunk of text
  + And scan from file function – which can read files (column-wise) with different types of data in each column
* Side note: e-num controls case structure and tab control (using property node)
* Side note: can set single element default values > right click > data operations

**Running loops concurrently (and synchronizing events with notifiers)**

* Open /examples and functions/examples/week6\_while\_loops.vi
* On BD:
  + Loops 1, 2 run at user-defined intervals and just output iteration number
  + Loop 5 does the same, but runs continuously (no wait timer), and continuously reads loop 3 iteration via a **local variable**
  + Loop 3 is like loops 1,2, but, it also activates a **notifier** when the user clicks the OK button
  + Loop 4 plots the iteration count from loops 1-3
* Run the VI and notice what happens when you press the Send /OK botton
* To demonstrate:
  + Multiple loops can run concurrently
    - May be a way to save computational power, if different components of the VI can run less frequently than others
    - Or allows the user to save / plot etc. at a different frequency than information is generated
  + Loops can be controlled by either one or multiple stop signs
  + That information can be passed between loops using notifiers and local variables
  + Show the difference between local variables and notifiers
    - Local variables are always searching
    - Notifiers only activate when signalled
      * Save computational power
      * Enables synchronization

**Exercises**

Exercise 1:

To week4\_exercise4 / exercise5, add functionality to **calibrate** your numeric signal using an external file.

To do this, create a sub-VI that reads the calibration file (/examples & functions/examples/ week6\_numeric\_calibration.csv), and spits out the calibration coefficients. Supply a path name / location for the data file (consider how you will set up the browse options of the path control variable) that can be controlled from the main VI. Make the output a **cluster** containing: 1) the 2 calibration coefficients (stored in a 2x1 array), 2) the date of calibration, and 3) the form of the calibration equation.

Save the VI to your LabVIEW library as week6\_read\_calib\_subVI.vi.

Use the calibration coefficients to modify the number before it is plotted.

Plot both the calibrated and un-calibrated values.

Save As week6\_exercise1.vi to your LabVIEW library.

*Some tips:*

* Consider hard-wiring a string constant onto your data filenames depending on the file save mechanism you are using (e.g. Save to Measurement File 🡪 .lvm vs Save to CSV 🡪 .csv)
* Use the unbundle by name to identify the elements of the calibration file cluster.

Exercise 2:

Add a way to save comments to your growing data file. Use a front panel string control to write a data comment which will be saved to your growing data file at the end of each line of data.

Remember to modify both saving methods.

Save As week6\_exercise2.vi to your LabVIEW library.

*Some tips:*

* Use a flat sequence to save the data with comments, and then clear the comments string.

Exercise 3:

Imagine a scenario where you would like to save a subset of your data to a secondary location (for example, to be shared on a ship’s server, sent via a satellite transmission, or down-sample the data resolution). Modify the exercise2 VI to achieve this, and to read out the data from the backup file:

Every 10 iterations, save the most recent data to a **separate data file.** Use a front panel path control to identify the location and name of that backup file.

Create a **second while loop** to then read that file and update an array containing all of the file’s data every 20 seconds.

Save As week6\_exercise3.vi to your LabVIEW library.

*Some tips:*

* Consider how you will set the input filename for the function that reads out the data, and whether your function will be reading the data one line at a time, or the entire array at once.
* You may also need to “reset” the measurement read function.
* Think about how you will setup and grow your array.

Exercise 4:

Add a mechanism to **save metadata in your VI.** The metadata should at least contain the calibration information and today’s date. (use /examples & functions/examples/week6\_metadata.txt as an example)

Save the data as a .txt file.

Use a front panel control to identify where the file will be saved.

Save As week6\_exercise4.vi to your LabVIEW library.

*Some tips:*

* Consider if you will add these elements inside or outside of the while loops.

**First class next year:**

Some common errors

Intro to error handling

Reviewing what we’ve learned so far

Preview what’s to come