Modalities

- Example
  - code for matrix multiplication in `/afs/ictp.it/home/o/obrovko/public/num_I/timing`
- Submit
  - by Wednesday, Nov 15, 2017
  - send in the code(s) and the gnuplot script(s)
- Lessons to learn
  - know what operations matter
  - try to think in terms of memory structure (multi-dimensional arrays)
- Optional
  - the part about the sorting algorithm is optional (bonus points)
  - to save data for plotting either write to different files in Fortran or use output redirection in bash
  - if you prefer, you can do the assignment in a programming language of your choice
    - C/C++, Fortran, Python, Java, Perl, PHP, Matlab, R
    - as long as you can install an appropriate compiler on an ICTP machine

come find me f you have question: LB.226 | obrovko@ictp.it

Timing Assignment #1

- Timing operations 1D arrays
  - times are very short, so do it with arrays of many elements (f.e. 1e5)
  - assignment (integer array) `varInt   = 12345`
  - assignment (real array) `varReal  = 12.3456`
  - assignment (double array) `varDouble   = 12.3456_dp`
  - summation (1D array) `varDouble   = 12.3456_dp + 65.4321_dp`
  - multiplication (1D array) `array_c = array_a * array_b`
- Timings tor 2D arrays
  - row-major `(i,j)`

```
do i = 1:N
    do j = 1:N
    array_c(i,j) = array_a(i,j) * array_b(i,j)
    enddo
enddo
```

  - column-major `(j,i)`

```
do j = 1:N
    do i = 1:N
        array_c(i,j) = array_a(i,j) * array_b(i,j)
    enddo
enddo
```

  - built-in 2D array multiplication `array_c = array_a * array_b`

Timing Assignment #2

- Operations' complexity
  - test scaling of operations (assignment, summation, multiplication row-major and column-major)

```
do n=500, 10000, 500    (adjust according to times from assignment #1)
   allocate(matA(n,n))
   allocate(matB(n,n))
   allocate(matC(n,n))
   do rep = 1, 10       (adjust according to times from assignment #1)
      call cpu_time(t1)
      i = 1, n
         j = 1, n
            matC(i,j) = matA(i,j) * mat(i,j)
         enddo
      enddo
      call cpu_time(t2)
      times(rep) = t2-t1
   enddo
   write (outfile,'(I5,F11.3,F11.3)') n, t_mean*1d3, t_stderr*1d3   (time in ms)
enddo
```

  - plot time vs n in gnuplot
- Timing sorting routine
  - time the sorting routine for arrays of different lengths (1,10001 in steps of 100)
  - plot time needed to sort vs n
  - randomize array before each sorting