Fortran/C on HCC

This quick start demonstrates how to implement a Fortran/C program on HCC supercomputers. The sample codes and submit scripts can be downloaded from <serial_dir.zip>.

Login to a HCC Cluster (Tusker or Sandhills)

Log in to a HCC cluster through PuTTY (For Windows Users) or Terminal (For Mac/Linux Users) and make a subdirectory called serial_dir under the $WORK directory.

```
$ cd $WORK
$ mkdir serial_dir
```

In the subdirectory serial_dir, save all the relevant Fortran/C codes. Here we include two demo programs, demo_f_serial.f90 and demo_c_serial.c, that compute the sum from 1 to 20.

```
demo_f_serial.f90

Program demo_f_serial
  implicit none
  integer, parameter :: N = 20
  real*8 w
  integer i
  common/sol/ x
  real*8 x
  real*8, dimension(N) :: y

  do i = 1,N
    w = i*1d0
    call proc(w)
    y(i) = x
    write(6,*) 'i,x = ', i, y(i)
  enddo
  write(6,*) 'sum(y) =',sum(y)
Stop
End Program

Subroutine proc(w)
  real*8, intent(in) :: w
  common/sol/ x
  real*8 x

  x = w

Return
End Subroutine
```
```c
#include <stdio.h>

double proc(double w){
    double x;
    x = w;
    return x;
}

int main(int argc, char* argv[]){
    int N=20;
    double w;
    int i;
    double x;
    double y[N];
    double sum;
    for (i = 1; i <= N; i++){
        w = i*1e0;
        x = proc(w);
        y[i-1] = x;
        printf("i,x= %d %lf\n", i, y[i-1]) ;
    }
    sum = 0e0;
    for (i = 1; i<= N; i++){
        sum = sum + y[i-1];
    }
    printf("sum(y)= %lf\n", sum);
    return 0;
}
```

Compiling the Code

The compiling of a Fortran/C++ code to executable is usually done behind the scene in a Graphical User Interface (GUI) environment, such as Microsoft Visual Studio. In a HCC cluster, the compiling is done explicitly by first loading a choice compiler and then executing the corresponding compiling command. Here we will use the GNU Compiler Collection, `gcc`, for demonstration. Other available compilers such as `intel` or `pgi` can be looked up using the command line `module avail`. Before compiling the code, make sure there is no dependency on any numerical library in the code. If invoking a numerical library is necessary, contact a HCC specialist (hcc-support@unl.edu) to discuss implementation options.

```
$ module load compiler/gcc/7.1
$ gfortran demo_f_serial.f90 -o demo_f_serial.x
$ gcc demo_c_serial.c -o demo_c_serial.x
```

The above commands load the `gcc` compiler and use the compiling commands `gfortran` or `gcc` to compile the codes to `.x` files (executables).
Creating a Submit Script

Create a submit script to request one core (default) and 1-min run time on the supercomputer. The name of the main program enters at the last line.

```
submit_f.serial
#!/bin/sh
#SBATCH --mem-per-cpu=1024
#SBATCH --time=00:01:00
#SBATCH --job-name=Fortran
#SBATCH --error=Fortran.%J.err
#SBATCH --output=Fortran.%J.out

module load compiler/gcc/4.9
./demo_f_serial.x
```

```
submit_c.serial
#!/bin/sh
#SBATCH --mem-per-cpu=1024
#SBATCH --time=00:01:00
#SBATCH --job-name=C
#SBATCH --error=C.%J.err
#SBATCH --output=C.%J.out

module load compiler/gcc/4.9
./demo_c_serial.x
```

Submit the Job

The job can be submitted through the command `sbatch`. The job status can be monitored by entering `squeue` with the `-u` option.

```
$ sbatch submit_f.serial
$ sbatch submit_c.serial
$ squeue -u <username>
```

Sample Output

The sum from 1 to 20 is computed and printed to the `.out` file (see below).
### Fortran.out

<table>
<thead>
<tr>
<th>i, x =</th>
<th>1</th>
<th>1.0000000000000000</th>
</tr>
</thead>
<tbody>
<tr>
<td>i, x =</td>
<td>2</td>
<td>2.0000000000000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>3</td>
<td>3.0000000000000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>4</td>
<td>4.0000000000000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>5</td>
<td>5.0000000000000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>6</td>
<td>6.0000000000000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>7</td>
<td>7.0000000000000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>8</td>
<td>8.0000000000000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>9</td>
<td>9.0000000000000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>10</td>
<td>10.0000000000000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>11</td>
<td>11.0000000000000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>12</td>
<td>12.0000000000000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>13</td>
<td>13.0000000000000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>14</td>
<td>14.0000000000000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>15</td>
<td>15.0000000000000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>16</td>
<td>16.0000000000000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>17</td>
<td>17.0000000000000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>18</td>
<td>18.0000000000000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>19</td>
<td>19.0000000000000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>20</td>
<td>20.0000000000000000</td>
</tr>
</tbody>
</table>

**sum(y) = 210.00000000000000**

### C.out

<table>
<thead>
<tr>
<th>i, x =</th>
<th>1</th>
<th>1.000000</th>
</tr>
</thead>
<tbody>
<tr>
<td>i, x =</td>
<td>2</td>
<td>2.000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>3</td>
<td>3.000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>4</td>
<td>4.000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>5</td>
<td>5.000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>6</td>
<td>6.000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>7</td>
<td>7.000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>8</td>
<td>8.000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>9</td>
<td>9.000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>10</td>
<td>10.000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>11</td>
<td>11.000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>12</td>
<td>12.000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>13</td>
<td>13.000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>14</td>
<td>14.000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>15</td>
<td>15.000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>16</td>
<td>16.000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>17</td>
<td>17.000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>18</td>
<td>18.000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>19</td>
<td>19.000000</td>
</tr>
<tr>
<td>i, x =</td>
<td>20</td>
<td>20.000000</td>
</tr>
</tbody>
</table>

**sum(y) = 210.00000000000000**