RUNNING C, C++, FORTRAN, PYTHON, AND R CODE- SERIAL MODE

Table of Contents

1. **OVERVIEW** 1
2. **CLONE THE GITHUB REPOSITORY** 2
3. **COMPILING AND RUNNING A SAMPLE SERIAL C PROGRAM** 3
4. **COMPILING AND RUNNING A SAMPLE C++ PROGRAM** 4
5. **COMPILING AND RUNNING A SAMPLE SERIAL FORTRAN PROGRAM** 6
6. **RUNNING A SAMPLE PYTHON PROGRAM USING PYTHON 3** 8
7. **RUNNING A SAMPLE R SCRIPT** 9

1. OVERVIEW

This section of the user-guide covers the steps to run sample serial and parallel code in C, C++, Fortran, Python, and R using both interactive and batch job submission modes. Where relevant, separate steps for using Intel and GNU compilers are covered.

Please note that **all code should be run only on the compute nodes** either interactively or in batch mode. Note that when a batch job is submitted on the login node, the batch job script specifies that the commands are to be run on the compute nodes. **Please DO NOT run any code on the login nodes**. Acceptable use of login nodes is as follows: installing code, file transfer, file editing, and job submission and monitoring.

Note: The examples provided in this document include the loading of specific Linux environment modules. These modules adjust your shell configuration, such as the $PATH environment variable, to accommodate particular software packages. Further details and information regarding the management of Linux modules, such as loading, unloading, swapping, and other operations, can be found here: ModuleEnvironments < ARC < UTSA Research Support Group.

2. CLONE THE GITHUB REPOSITORY

If you want to get a copy of all the programs and scripts used in the examples shown in this document, you can clone the GitHub repository for the examples using the following command:

```
git clone https://github.com/ritua2/documentation
```

If you cloned the GitHub repository, you can switch to the documentation directory with the following command to find the scripts and programs referred to throughout the document:

```
[login001]$ cd documentation
```

The **documentation** directory structure is shown below (split across four columns):

```
.
└── documentation
    └── serial_jobs
        ├── serialc
        │   ├── GNU
        │   │   ├── hello_world.c
        │   │   └── hello_world.slurm
        │   └── Intel
        │       ├── hello_world.c
        │       └── hello_world.slurm
        ├── serialcpp
        │   ├── GNU
        │   │   ├── hello_world.cpp
        │   │   └── hello_world.slurm
        │   └── Intel
        │       ├── hello_world.cpp
        │       └── hello_world.slurm
        ├── serialf
        │   ├── GNU
        │   │   ├── hello_world.f90
        │   │   └── hello_world.slurm
        │   └── Intel
        │       ├── hello_world.f90
        │       └── hello_world.slurm
        ├── serialpy
        │   ├── hello_world.py
        │   └── hello_world.slurm
        └── serialR
            ├── hello_world.r
            └── hello_world.slurm
```

```
.
└── documentation
    └── cuda_jobs
        ├── cudac
        │   ├── hello_world.cu
        │   └── hello_world.slurm
        ├── cudacpp
        │   ├── hello_world.cu
        │   └── hello_world.slurm
        └── cudaf
            ├── hello_world.cuf
            └── hello_world.slurm
```

```
.
└── documentation
    ├── mpi_jobs
    │   ├── mpic
    │   │   ├── hello_world.c
    │   │   └── hello_world.slurm
    │   ├── mpicpp
    │   │   ├── hello_world.cpp
    │   │   └── hello_world.slurm
    │   └── mpif
    │       ├── hello_world.f90
    │       └── hello_world.slurm
    └── openmp_jobs
        ├── openmpc
        │   ├── GNU
        │   │   ├── hello_world.c
        │   │   └── hello_world.slurm
        │   └── Intel
        │       ├── hello_world.c
        │       └── hello_world.slurm
        ├── openmpcpp
        │   ├── GNU
        │   │   ├── hello_world.cpp
        │   │   └── hello_world.slurm
        │   └── Intel
        │       ├── hello_world.cpp
        │       └── hello_world.slurm
        └── openmpf
            ├── GNU
            │   ├── hello_world.f90
            │   └── hello_world.slurm
            └── Intel
                ├── hello_world.f90
                └── hello_world.slurm
```

```
.
└── documentation
    ├── GNUparallel
    │   ├── hello_world.c
    │   ├── hello_world.slurm
    │   ├── myout.txt
    │   └── PrimeFactorEx
    │       ├── paramfile.txt
    │       ├── prime_fac.c
    │       ├── prime_fac_rf.slurm
    │       └── prime_fac.slurm
    └── multiExes
        ├── mpi_multiexes
        │   ├── bye_world.c
        │   ├── hello_world.c
        │   └── hello_world.slurm
        └── serial
            ├── hello_world.c
            └── hello_world.slurm
```

When you switch to the subdirectories within the **documentation** folder, the Slurm job script files are available with the *.slurm extension.

If you do not want to clone the aforementioned GitHub repository, you should be able to copy the code shown in the listings into files with appropriate names and file extensions.

3. COMPILING AND RUNNING A SAMPLE SERIAL C PROGRAM

A sample C program is shown in Listing 1. This program will print "`Hello World!!`" to standard output.

```
#include <stdio.h>
int main(){
 printf("Hello World!!");
 return 0;
}
```

Listing 1: Sample C program - hello_world.c (documentation/serial_jobs/serialc/GNU/hello_world.c)

If you would like to compile the C example using the GNU C compiler, you can run the following commands:

```
[login001]$ ml gnu8/8.3.0
[login001]$ gcc -o hello_world hello_world.c
```

If you would like to compile the C example using the Intel OneAPI, you can run the following commands:

```
[login001]$ ml intel/oneapi/2021.2.0
[login001]$ icc -o hello_world hello_world.c
```

The executable `hello_world` can be run either in a batch mode using a Slurm batch job-script or interactively on a compute node.

Running the Executable in Interactive-Mode: The executable can be run interactively on a compute node using the following set of commands and the output will be displayed on the terminal:

```
[login001]$ srun -p compute1 -n 1 -t 00:05:00 --pty bash
[c001]$ ./hello_world
Hello World!!
```

If you are currently on a compute node and would like to switch back to the login node then enter the `exit` command as follows:

```
[c001]$ exit
```

Running the Executable in Batch-Mode: A sample Slurm batch job-script to run the executable named `hello_world` is shown in Listing 2. This batch script corresponds to the serial program named `hello_world.c`. This script should be run from a login node.

```
#!/bin/bash
#SBATCH -J  hello_world_c
#SBATCH -o  hello_world.txt
#SBATCH -p compute1
#SBATCH -t 00:02:00
#SBATCH -N 1
#SBATCH -n 1

./hello_world #the executable name is obtained after compiling
the program
```

Listing 2: Batch Job Script for C code – hello_world.slurm
(documentation/serial_jobs/serialc/GNU/hello_world.slurm)

The job-script shown in Listing 2 can be submitted as follows:

```
[login001]$ sbatch hello_world.slurm
```

The output from the Slurm batch-job shown in Listing 2 can be checked by opening the output file as follows:

```
[login001]$ cat hello_world.txt
Hello World!!
```

4. COMPILING AND RUNNING A SAMPLE C++ PROGRAM

A sample C++ program is shown in Listing 3. This program prints "`Hello World`" to standard output.

```
# include <iostream>
using namespace std;

int main(){
   cout<<"Hello World!!";
   return 0;
}
```

Listing 3: Sample C++ program - hello_world.cpp (documentation/serial_jobs/serialcpp/GNU/hello_world.cpp)

If you would like to compile the C++ example using the GNU C++ compiler, then you can run the following commands:

```
[login001]$ ml gnu8/8.3.0
[login001]$ g++ -o hello_world hello_world.cpp
```

If you would like to compile the C++ example using the Intel OneAPI, then you can run the following commands:

```
[login001]$ ml intel/oneapi/2021.2.0
[login001]$ icpc -o hello_world hello_world.cpp
```

The executable `hello_world` can be run either in a batch mode using a Slurm batch job-script or interactively on a compute node.

Running the Executable in Interactive-Mode: The executable can be run interactively on a compute node using the following set of commands and the output will be displayed directly on the terminal:

```
[login001]$ srun -p compute1 -n 1 -t 00:05:00 --pty bash
[c001]$ ./hello_world
Hello World!!
```

If you are currently on a compute node and would like to switch back to the login node then enter the exit command as follows:

```
[c001]$ exit
```

Running the Executable in Batch-Mode: A sample Slurm batch job-script to run the executable named `hello_world` is shown in Listing 4. This batch script corresponds to the serial program named `hello_world.cpp`. This script should be run from a login node.

```
#!/bin/bash
#
#SBATCH -J hello_world_cpp
#SBATCH -o hello_world.txt
#SBATCH -p compute1
#SBATCH -t 00:05:00
#SBATCH -N 1
#SBATCH -n 1

./hello_world #the executable name is obtained after compiling
the program
```

Listing 4: Batch Job Script for C++ code – hello_world.slurm (documentation/serial_jobs/serialcpp/GNU/hello_world.slurm)

The job-script shown in Listing 4 can be submitted as follows:

```
[login001]$ sbatch hello_world.slurm
```

The output from the Slurm batch-job shown in Listing 4 can be checked by opening the output file as follows:

```
[login001]$ cat hello_world.txt
 Hello World!!
```

5. COMPILING AND RUNNING A SAMPLE SERIAL FORTRAN PROGRAM

A sample Fortran program is shown in Listing 5. This program will print "`Hello World!!`" to the standard output.

```
program hello
 print *, 'Hello World!!'
end program hello
```

Listing 5: Sample Fortran Program - hello_world.f90 (documentation/serial_jobs/serialf/GNU/hello_world.f90)

If you would like to compile the Fortran example using the GNU Fortran compiler, you can run the following commands:

```
[login001]$ ml gnu8/8.3.0
[login001]$ gfortran -o hello_world hello_world.f90
```

If you would like to compile the Fortran example using the Intel OneAPI, you can run the following commands:

```
[login001]$ ml intel/oneapi/2021.2.0
[login001]$ ifort -o hello_world hello_world.f90
```

The executable `hello_world` can be run either in a batch mode using a Slurm batch job-script or interactively on a compute node.

Running the Executable in Interactive-Mode: The executable can be run interactively on a compute node using the following set of commands and the output will be displayed on the terminal:

```
[login001]$ srun -p compute1 -n 1 -t 00:05:00 --pty bash
[c001]$ ./hello_world
Hello World!!
```

If you are currently on a compute node and would like to switch back to the login node then enter the `exit` command as follows:

```
[c001]$ exit
```

Running the Executable in Batch-Mode: A sample Slurm batch job-script to run the executable named `hello_world` is shown in Listing 6. This batch script corresponds to the serial program named `hello_world.f90`. This script should be run from a login node.

```
#!/bin/bash
#SBATCH -J hello_world_f90
#SBATCH -o hello_world.txt
#SBATCH -p compute1
#SBATCH -t 00:05:00
#SBATCH -N 1
#SBATCH -n 1

./hello_world #the executable name is obtained after compiling
the program
```

Listing 6: Batch Job Script for Fortran code – hello_world.slurm
(documentation/serial_jobs/serialf/GNU/hello_world.slurm)

The job-script shown in Listing 6 can be submitted as follows:

```
[login001]$ sbatch hello_world.slurm
```

The output from the Slurm batch-job shown in Listing 6 can be checked by opening the output file as follows:

```
[login001]$ cat hello_world.txt
Hello World!
```

6. RUNNING A SAMPLE PYTHON PROGRAM USING PYTHON 3

A sample Python program is shown in Listing 7. This program will print "`Hello World!!`" to standard output.

```
print('Hello World!!')
```

Listing 7: Sample Python program - hello_world.py
(documentation/serial_jobs/serialpy/hello_world.py)

The Python program can be run either in batch mode using a Slurm batch job-script or interactively on a compute node.

Running the Code in Interactive-Mode: The program can be run interactively on a compute node using the following set of commands and the output will be displayed on the terminal:

```
[login001]$ srun -p compute1 -n 1 -t 00:05:00 --pty bash
[c001]$ python3 hello_world.py
Hello World!!
```

If you are currently on a compute node and would like to switch back to the login node then enter the exit command as follows:

```
[c001]$ exit
```

Running the Code in Batch-Mode: A sample Slurm batch job-script to run `hello_world.py` in batch mode is shown in Listing 8. This script should be run from a login node.

```
#!/bin/bash
#SBATCH -J hello_world_py
#SBATCH -o hello_world.txt
#SBATCH -p compute1
#SBATCH -t 00:05:00
#SBATCH -N 1
#SBATCH -n 1

python3 hello_world.py #Python program_name
```

Listing 8: Batch Job Script for Python code – hello_world.slurm (documentation/serial_jobs/serialpy/hello_world.slurm)

The job-script shown in Listing 8 can be submitted as follows:

```
[login001]$ sbatch hello_world.slurm
```

The output from the Slurm batch-job shown in Listing 8 can be checked by opening the output file as follows:

```
[login001]$ cat hello_world.txt
Hello World!!
```

7. RUNNING A SAMPLE R SCRIPT

A sample R script is shown in Listing 9. This code will print "`Hello World!!`" to standard output.

```
print("Hello World!!")
```

Listing 9: Batch Job Script for R code – hello_world.r
(documentation/serial_jobs/serialR/hello_world.r)

Add execute permissions to the R script file as follows:

```
[login001]$ chmod +x hello_world.r
```

This R program can now be run either in a batch mode using a Slurm batch job-script or interactively on a compute node.

Running the script in Interactive-Mode: The executable can be run interactively on a compute node using the following set of commands and the output will be displayed on the terminal:

```
[login001]$ srun -p compute1 -n 1 -t 00:05:00 --pty bash
[c001]$ ml R/4.1.0
[c001]$ Rscript hello_world.r
[1] "Hello World!!"
```

If you are currently on a compute node and would like to switch back to the login node then enter the `exit` command as follows:

```
[c001]$ exit
```

Running the script in Batch-Mode: A sample Slurm batch job-script to run `hello_world.r` in batch mode is shown in Listing 10. This script should be run from a login node.

```
#!/bin/bash
#SBATCH -J hello_world_r
#SBATCH -o hello_world.txt
#SBATCH -p compute1
#SBATCH -t 00:05:00
#SBATCH -N 1
#SBATCH -n 1

ml R/4.1.0
Rscript hello_world.r #R program_name
```

Listing 10: Batch Job Script for R code – hello_world.slurm
(documentation/serial_jobs/serialR/hello_world.slurm)

The job-script shown in Listing 10 can be submitted as follows:

```
[login001]$ sbatch hello_world.slurm
```

The output from the Slurm batch-job shown in Listing 10 can be checked by opening the output file as follows:

```
[login001]$ cat hello_world.txt
[1] "Hello World!!"
```