




## Running Parallel Programming Data-Acquisition Scripts from a Windows Powershell



**Oregon State University**  
Mike Bailey  
mjb@cs.oregonstate.edu



This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](https://creativecommons.org/licenses/by-nc-nd/4.0/).



Computer Graphics

mpb - March 9, 2023

## Change the NUMT and NUMTRIES to Global int Variables

Right now, if your code is using defined constants, like this:


```
#ifndef NUMT
#define NUMT          2
#endif

#ifndef NUMS
#define NUMS          32
#endif
```

Change it to use global variables, like this:

```
int NUMT = 2;

int NUMS = 32;
```



Computer Graphics

mpb - March 9, 2023

## argc and argv

When you write in C or C++, your *main* program, which is really a special function call, looks like this:

```
int main( int argc, char *argv[ ] )
{
    ...
}
```

These arguments describe what was entered on the command line used to run the program.

The **argc** is the number of arguments (the *arg Count*)


The **argv** is a list of *argc* character strings that were typed (the *arg Vector*). The name of the program counts as the 0<sup>th</sup> *argv* (i.e., *argv[0]*)

So, for example, when you type

```
ls -l
```

in a shell, the *ls* program sees *argc* and *argv* filled like this:

```
argc = 2
argv[0] = "ls"
argv[1] = "-l"
```



Computer Graphics

mpb - March 9, 2023

## argc and argv

So, if *NUMT* and *NUMTRIALS* are global int variables:

```
int NUMT = 2;
int NUMS = 32;
```

and you want to set them from the command line, like this:

```
.Jprog 1 64
```


Then, *inside your main program*, you would say this:

```
if( argc >= 2 )
    NUMT = atoi( argv[1] );

if( argc >= 3 )
    NUMS = atoi( argv[2] );
```

The if-statements guarantee that nothing bad happens if you forget to type values on the command line.

The *atoi* function converts a string into an integer ("ascii-to-integer"). If you ever need it, there is also an *atof* function for floating-point.



Computer Graphics


mpb - March 9, 2023

## shared( ) in the #pragma omp Line

Also, remember, since *NUMTRIALS* is a variable, it needs to be declared as *shared* in the *#pragma omp* line:

```
#pragma omp parallel for default(none) shared(NUMS,xcs,ycs,rs,tn) reduction(+:numHits)
```

*NUMT* does not need to be declared in this way because it is not used in the for-loop that has the *#pragma omp* in front of it.



Computer Graphics

mpb - March 9, 2023

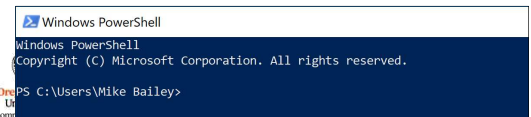
## Windows Powershell


Windows comes with a shell program called *Powershell*. It might not be as familiar to most of us as some of the Linux shells are (*bash*, *csh*), but it can still be used to run multiple combinations of your program parameters in one shot.

There are a number of ways to get Powershell running. Either:

- Click on the Microsoft icon. Then scroll down to **Windows Powershell** and run **Windows Powershell**.
- Shift right-click in the directory you want to work in and select **Open Powershell Window**.
- Hold down the Windows key and hit the 'x' key, then select **Windows Powershell**.

The resulting window should look like this:





Computer Graphics

mpb - March 9, 2023

## Change Directory to Where Your .exe File Lives

Then:

1. `cd` (change directory) to your home directory.
2. Then `cd` to the folder with your project
3. Then `cd` to the folder with your executable (\*.exe)

```
Windows PowerShell
PS C:\wjb\CS575\MonteCarlo\msvc\Debug>
```

The prompt will always tell you where you are in the file system.

## Running an Executable

So, if you have `cd`'ed to where your executable (.exe) file lives, you can run it from the command line like this:

```
Windows PowerShell
PS C:\wjb\CS575\MonteCarlo\msvc\Debug> ls
Directory: C:\wjb\CS575\MonteCarlo\msvc\Debug

Mode                LastWriteTime         Length Name
----                -
-a----         4/13/2020   8:28 PM           4714 MonteCarlo.exe
-a----         4/13/2020   8:28 PM       483468 MonteCarlo-ilk
-a----         4/13/2020   8:28 PM       626688 MonteCarlo.pdb

PS C:\wjb\CS575\MonteCarlo\msvc\Debug> ./MonteCarlo.exe
2 threads ; 50000 trials ; megatrials/sec = 57.42
PS C:\wjb\CS575\MonteCarlo\msvc\Debug>
```

## Running a Loop

But, here's the cool part. Type:

```
foreach ($i in 1, 2, 4 )
{
    foreach ($n in 1024, 2048, 4096)
    {
        ./MonteCarlo.exe $i $n
    }
}
```

followed by Enter:

```
Windows PowerShell
PS C:\wjb\CS575\MonteCarlo\msvc\Debug> foreach ($i in 1, 2, 4 )
{
    foreach ($n in 1024, 2048, 4096 )
    {
        ./MonteCarlo.exe $i $n
    }
}
1 threads ; 1024 trials ; megatrials/sec = 31.62
1 threads ; 2048 trials ; megatrials/sec = 30.65
1 threads ; 4096 trials ; megatrials/sec = 28.66
2 threads ; 1024 trials ; megatrials/sec = 62.25
2 threads ; 2048 trials ; megatrials/sec = 60.36
2 threads ; 4096 trials ; megatrials/sec = 58.16
4 threads ; 1024 trials ; megatrials/sec = 86.61
4 threads ; 2048 trials ; megatrials/sec = 90.55
4 threads ; 4096 trials ; megatrials/sec = 112.23
PS C:\wjb\CS575\MonteCarlo\msvc\Debug>
```

## Running a Loop from a File

You can also use a text editor like `notepad` or `notepad++` and put these lines into a file called, say, `loop.ps1` (ps1 is the Powershell file extension).

Then, you can run this script from Powershell just by typing it:

```
Windows PowerShell
PS C:\wjb\CS575\MonteCarlo\msvc\Debug> . loop.ps1
Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the
execution policy might expose you to the security risks described in the
about_Execution_Policies help topic at https://go.microsoft.com/fwlink/?LinkID=135170. Do you
want to change the execution policy?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): yes
PS C:\wjb\CS575\MonteCarlo\msvc\Debug> . loop.ps1
1 threads ; 1024 trials ; megatrials/sec = 31.62
1 threads ; 2048 trials ; megatrials/sec = 29.73
1 threads ; 4096 trials ; megatrials/sec = 29.95
2 threads ; 1024 trials ; megatrials/sec = 66.40
2 threads ; 2048 trials ; megatrials/sec = 48.59
2 threads ; 4096 trials ; megatrials/sec = 59.91
4 threads ; 1024 trials ; megatrials/sec = 124.50
4 threads ; 2048 trials ; megatrials/sec = 97.17
4 threads ; 4096 trials ; megatrials/sec = 79.08
PS C:\wjb\CS575\MonteCarlo\msvc\Debug>
```

Instead of printing these lines to the screen, you probably want to print them to a text file that can then be imported by Excel.

## Diverting Output to a File from Powershell

To divert the `print`'s to a file, do this:

```
./loop.ps1 > out.csv
```

or this:

```
./loop.ps1 1> out.csv
```

To divert *both* `print`'s and `fprint`(`stderr`,...) together, do this:

```
./loop.ps1 2>&1> out.csv
```