# Introduction to Julia Programming Language

Alper Kinaci, PhD
Sr. Computational Specialist
akinaci@northwestern.edu

Pascal Paschos, PhD
Sr. HPC Specialist
pascal.paschos@northwestern.edu

Research Computing Services

Northwestern
INFORMATION TECHNOLOGY

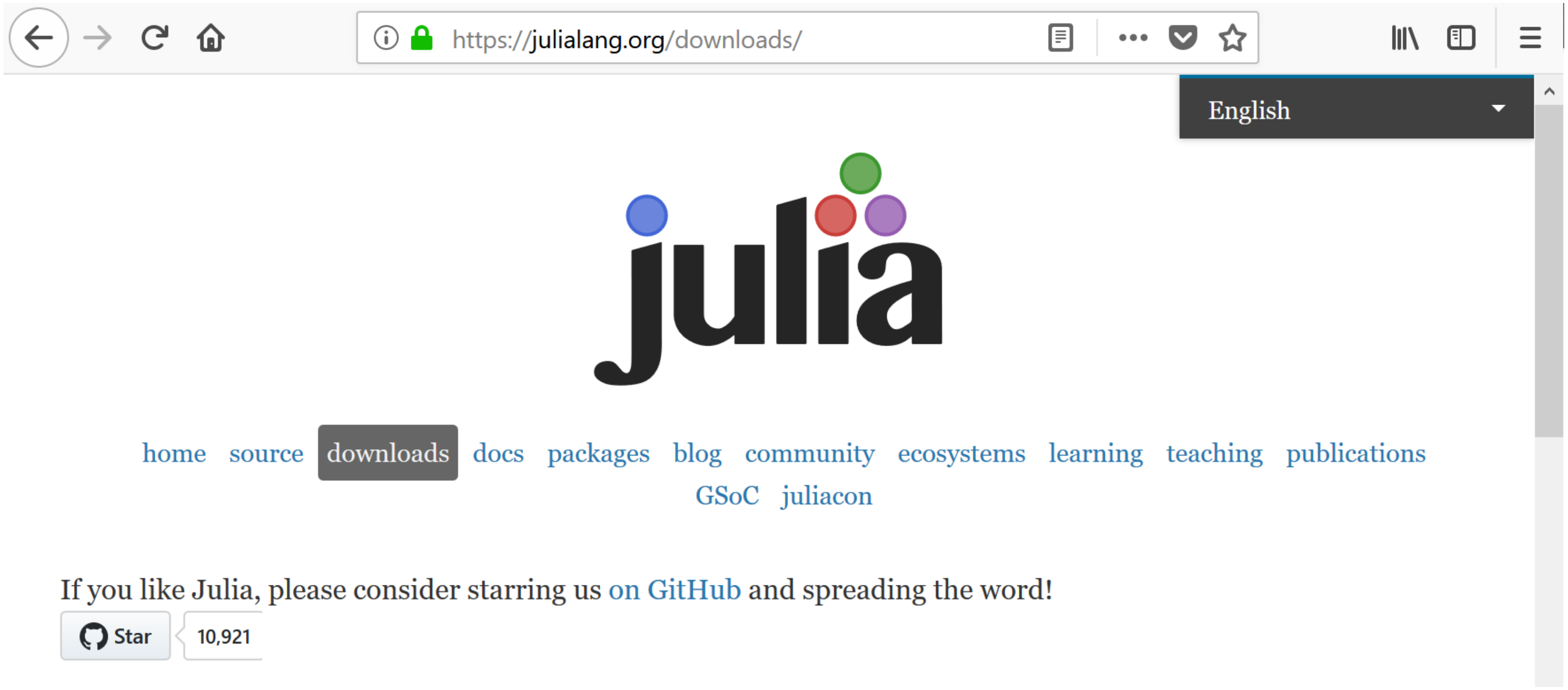
## Outline

- Introduction to Julia (short presentation)
- Live coding (using Jupyter notebook)
- Parallelization demonstration

# Julia Language

- “Walk like Python, run like C”
- Designed for ease-of-use (Python, MATLAB, R) and speed (C, C++) for high performance computational science
- Borrowing from Python, C, C++, MATLAB, Lisp, Perl, Lua, Ruby
- Open source, object-oriented, good for general-purpose programming
- Designed for parallelism and distributed computing
- Call for C & Fortran functions directly (no wrappers or special API)
- 20+ years younger than Python, still maturing in terms of available packages and extensibility

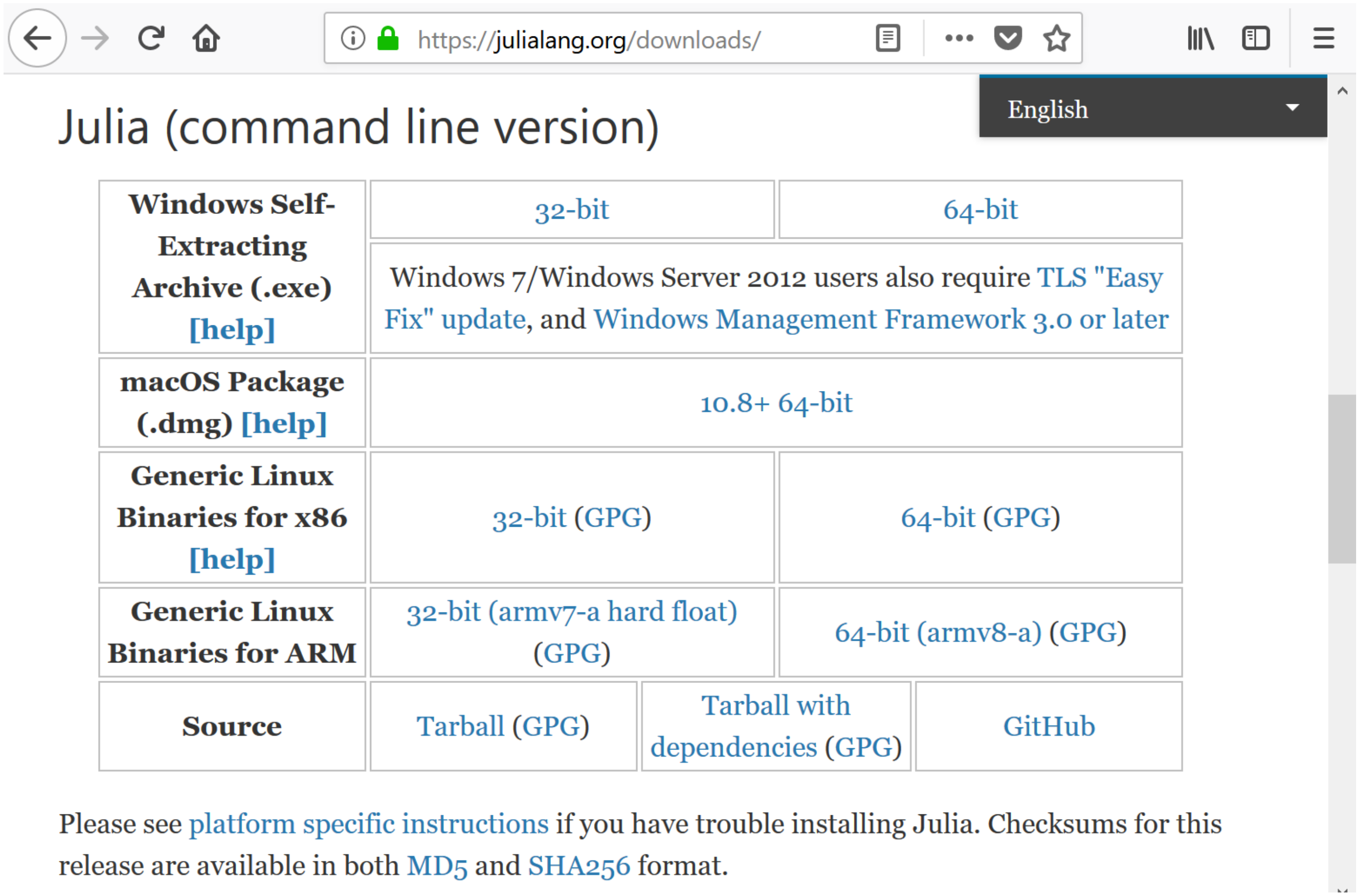
# How to Get Julia



If you like Julia, please consider starring us on GitHub and spreading the word!

https://julialang.org/downloads/

# How to Get Julia



https://julialang.org/downloads/

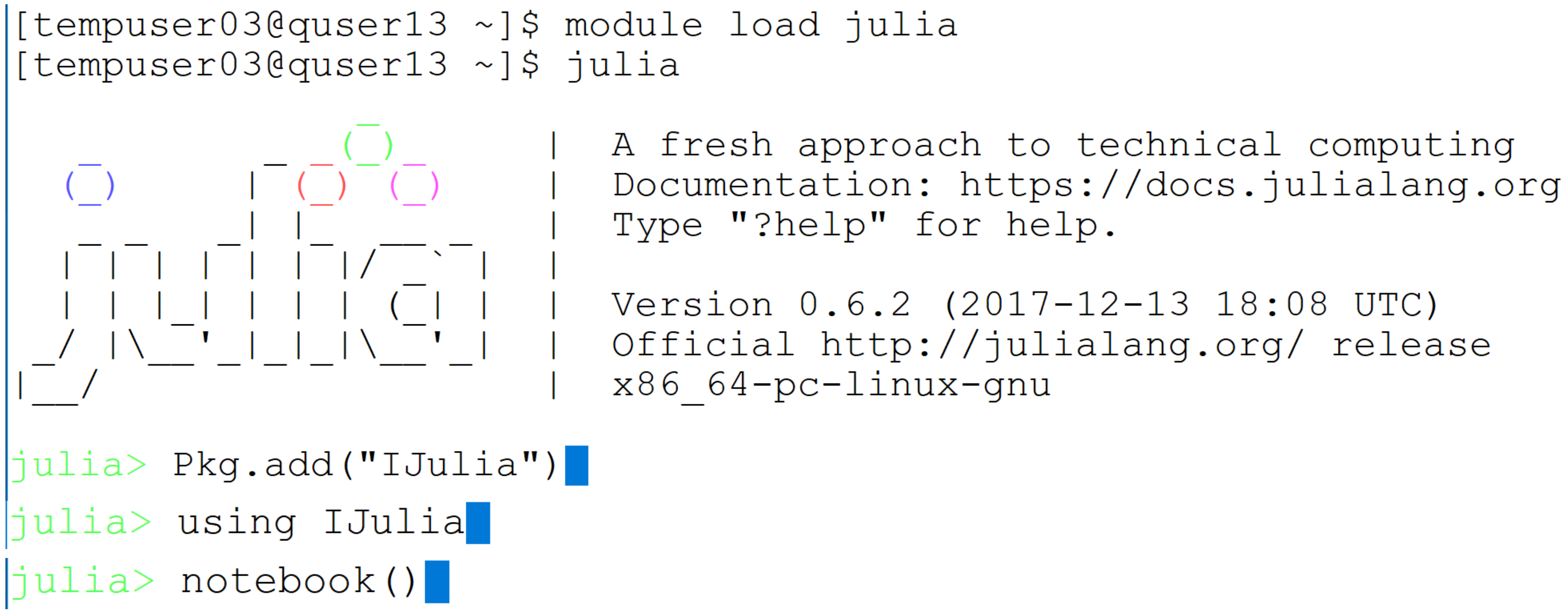
English

Julia (command line version)

| | | | |
|---|---|---|---|
| **Windows Self-Extracting Archive (.exe) [help]** | 32-bit | | 64-bit |
| | Windows 7/Windows Server 2012 users also require TLS "Easy Fix" update, and Windows Management Framework 3.0 or later | | |
| **macOS Package (.dmg) [help]** | 10.8+ 64-bit | | |
| **Generic Linux Binaries for x86 [help]** | 32-bit (GPG) | | 64-bit (GPG) |
| **Generic Linux Binaries for ARM** | 32-bit (armv7-a hard float) (GPG) | | 64-bit (armv8-a) (GPG) |
| **Source** | Tarball (GPG) | Tarball with dependencies (GPG) | GitHub |

Please see platform specific instructions if you have trouble installing Julia. Checksums for this release are available in both MD5 and SHA256 format.

# How to Get IJulia

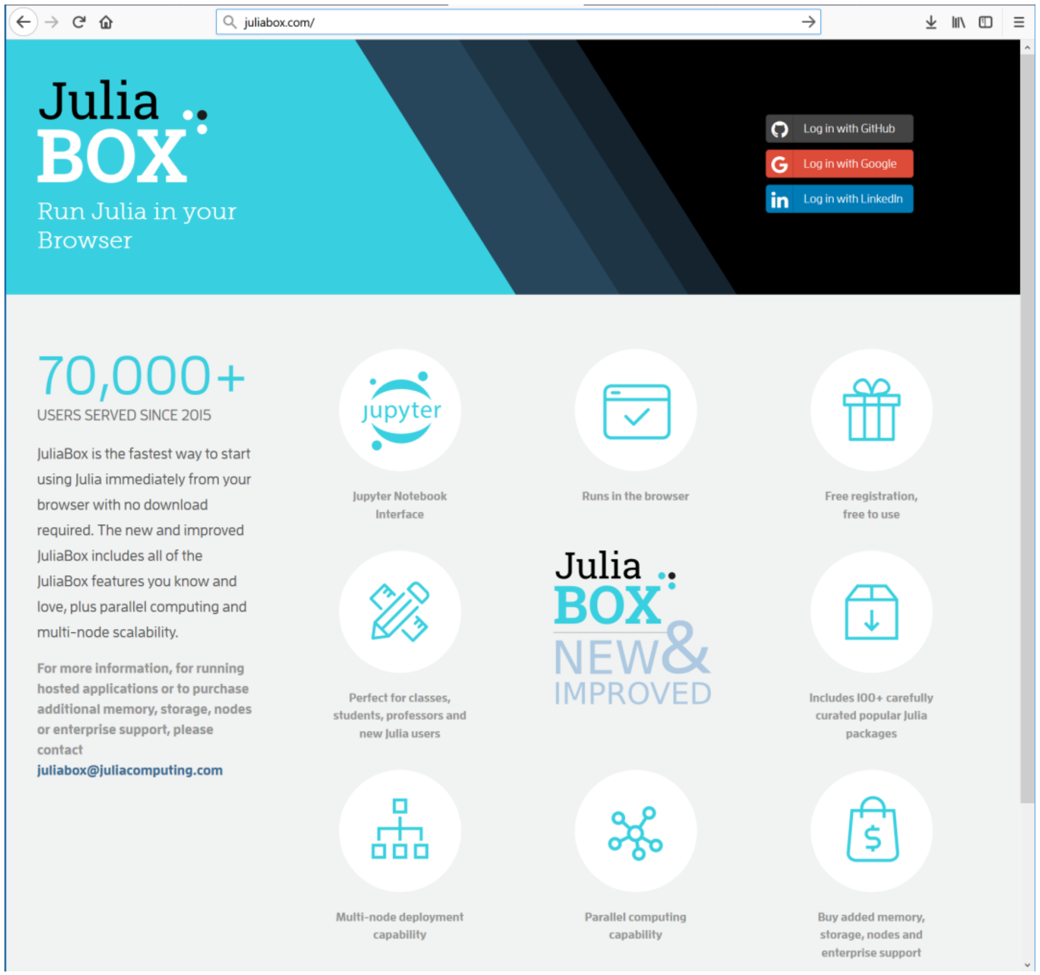


```
[tempuser03@quser13 ~]$ module load julia
[tempuser03@quser13 ~]$ julia
               _
   _       _ _(_)_     |  A fresh approach to technical computing
  (_)     | (_) (_)    |  Documentation: https://docs.julialang.org
   _ _   _| |_  __ _   |  Type "?help" for help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 0.6.2 (2017-12-13 18:08 UTC)
 _/ |\__'_|_|_|\__'_|  |  Official http://julialang.org/ release
|__/                   |  x86_64-pc-linux-gnu

julia> Pkg.add("IJulia")
julia> using IJulia
julia> notebook()
```

https://github.com/JuliaLang/IJulia.jl

# Instant Julia


juliabox.com/
Log in with GitHub
Log in with Google
Log in with LinkedIn
Jupyter Notebook Interface
Runs in the browser
Free registration, free to use
Perfect for classes, students, professors and new Julia users
Julia BOX NEW& IMPROVED
Includes 100+ carefully curated popular Julia packages
Multi-node deployment capability
Parallel computing capability
Buy added memory, storage, nodes and enterprise support

Julia BOX

Run Julia in your Browser

70,000+

USERS SERVED SINCE 2015

JuliaBox is the fastest way to start using Julia immediately from your browser with no download required. The new and improved JuliaBox includes all of the JuliaBox features you know and love, plus parallel computing and multi-node scalability.

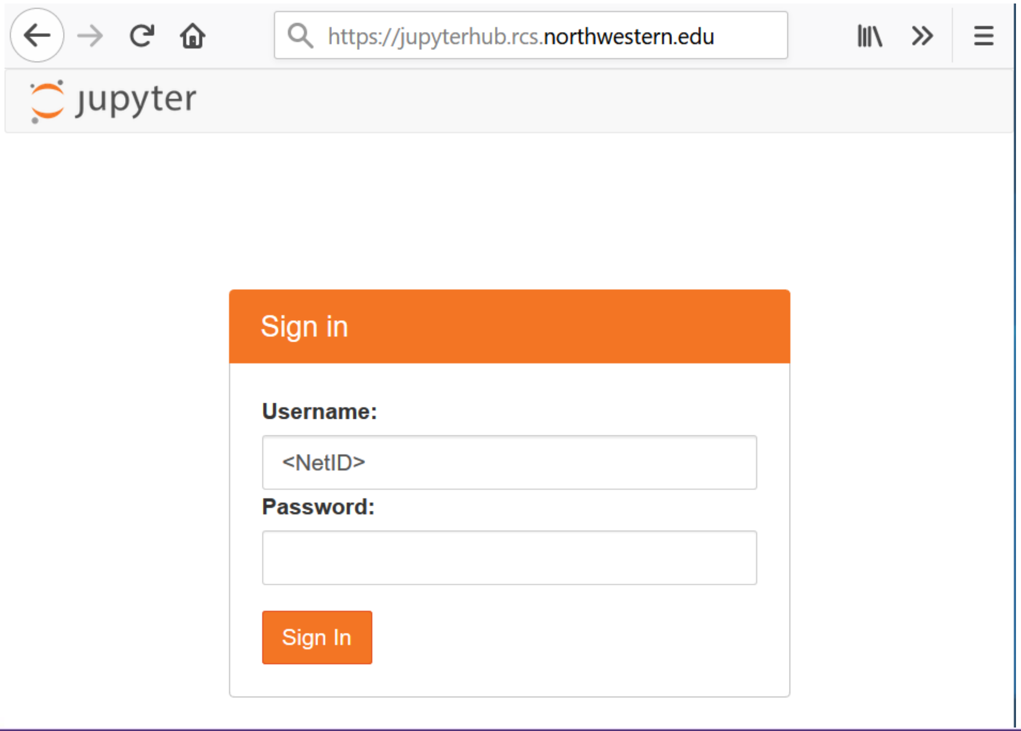
For more information, for running hosted applications or to purchase additional memory, storage, nodes or enterprise support, please contact juliabox@juliacomputing.com

https://juliabox.com

# Connect to Jupyter Hub

- The live coding part will use Jupyter notebook
- The Jupyter Hub will be available during the session

[https://jupyterhub.rcs.northwestern.edu](https://jupyterhub.rcs.northwestern.edu)


https://jupyterhub.rcs.northwestern.edu

## References

- Julia documentation [https://docs.julialang.org/en/stable/index.html]
- http://samuelcolvin.github.io/JuliaByExample/
- http://math.mit.edu/~stevenj/Julia-cheatsheet.pdf
- http://courses.csail.mit.edu/18.337/2017/
- Beginning Julia Programming, Sandeep Nagar, 2017[https://link.springer.com/book/10.1007%2F978-1-4842-3171-5]

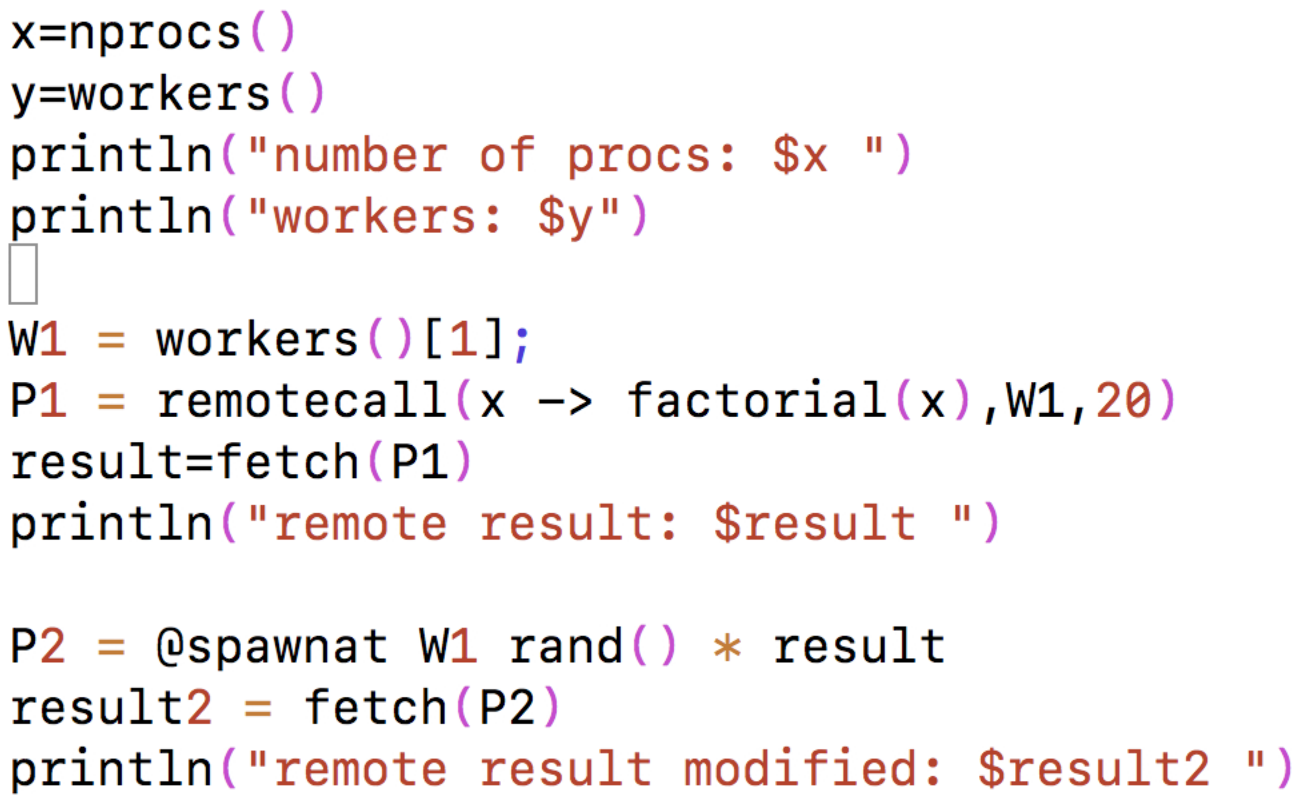
## Parallel Computing in Julia

- We will be demonstrating two ways of using Julia for parallel computing – We will not discuss the Coroutine method or native multithreading as of ver. 0.6
  - Native distributed multiprocessing. Limited to a single machine (node)
  - Julia MPI implementation - Multinode extension

# Native Distributed Multiprocessing

- You can invoke Julia as: julia –p (n-1) to launch (n-1) workers + master
- Caution. Significant changes in the syntax between current and earlier versions
- One sided communication to remote objects with distributed memory allocation
- Use of functions and macros to launch tasks and collect results, e.g.
  - remotecall(func,id,args): launch a task to worker id
  - fetch(value_of_remotecall): return remote calculation to master
  - @everywhere : launch to all – including master
  - @spawnat : evaluates on remote
  - @parallel: automatic loop parallelization

# Native Distributed Multiprocessing



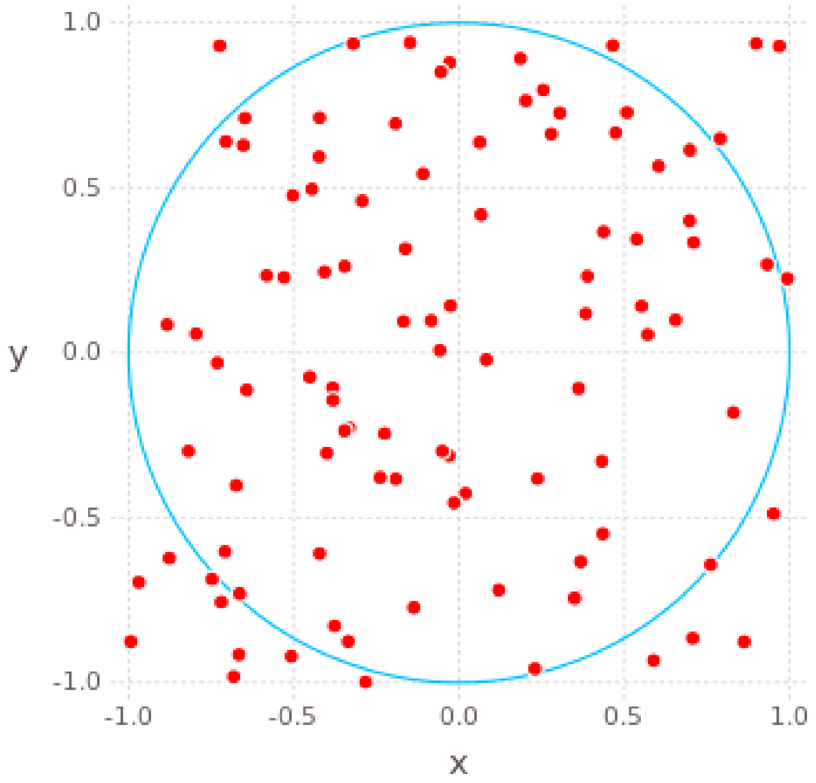
```
x=nprocs()
y=workers()
println("number of procs: $x ")
println("workers: $y")

W1 = workers()[1];
P1 = remotecall(x -> factorial(x),W1,20)
result=fetch(P1)
println("remote result: $result ")

P2 = @spawnat W1 rand() * result
result2 = fetch(P2)
println("remote result modified: $result2 ")
```

# Parallel Pi in Julia

π = 4 x fraction of points in circle



```
function findpi(n)
          inside = 0
          for i = 1:n
                x, y = rand(2) * 2 - 1
                if (x^2 + y^2 <= 1)
                      inside +=1
                end
          end
          pi = 4.0 * inside / n
          println("pi: $pi")
      end


function parallel_findpi(n)
          inside = @parallel (+) for i = 1:n
                x, y = rand(2) * 2 - 1
                x^2 + y^2 <= 1 ? 1 : 0
          end
          pi = 4.0 * inside / n
          println("pi: $pi")
      end
x=nprocs()
println("number of procs : $x")
if x == 1
        @time findpi(100000000)
else
        @time parallel_findpi(100000000)
end
```

# MPI with Julia

- **MPI** is one of Julia’s packages.
- Restores communication to MPI standard allowing Julia tasks to be distributed over a *network* of computers (nodes), aka cluster
- Remarkably similar to Python’s mpi4py
- **julia> Pkg.add(“MPI”)**
- User must supply the MPI wrappers, e.g. mpirun
- On Quest:

mpirun –np <N> julia <Julia_mpi_code>.jl