Introduction to Scientific Computing (BSR1015)

Assigned March 21st, Due April 4th

Recursively copy /sc/orga/scratch/costaa03/bash\_final folder to your home directory and cd to it. There will be 2 sub-directories: dir1 and dir2, and script, ttt.sh.

1. Consider the following script called ttt.sh:

#!/bin/bash

list=$1

for i in $list

do

echo $i

done

In this form it works thusly:

$ ./ttt.sh 1,2,3,4,5

1,2,3,4,5

Modify the “list=$1” line so that it works like the below (Hint: process substitution, stream editing).

Just change the line, do not add any other line/s; and do not use the IFS specification:

$ ./ttt.sh 1,2,3,4,5

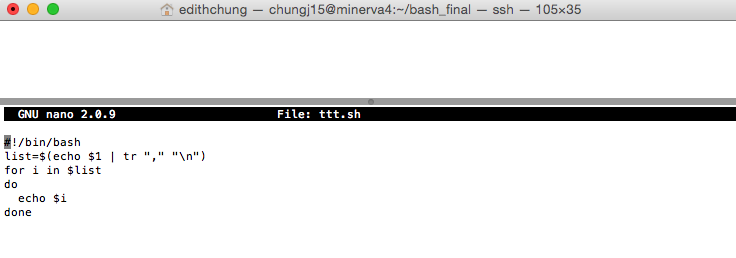
1

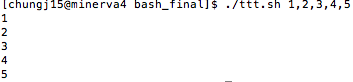
2

3

4

5

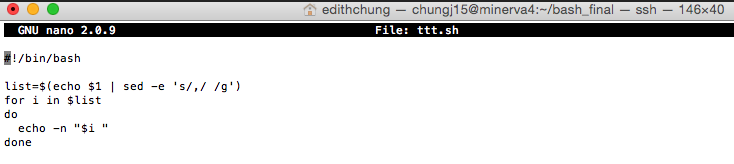




Now, modify the script so that it now works like:

$ ./ttt.sh 1,2,3,4,5

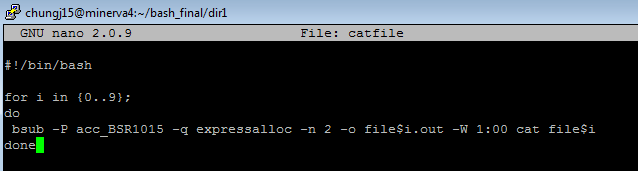
1 2 3 4 5

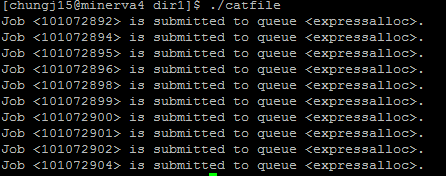


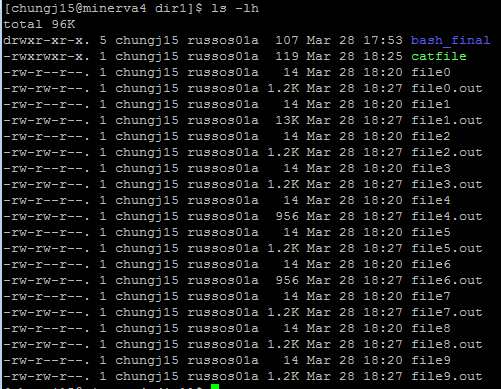
1. Write a script or just a bsub command line that will submit a job for each file in dir1 to the expressalloc queue that simply “cat”s the file. The output file (stdout) for each job should be the <filename>.out. That is, the first job should execute the command:

cat file1

and the output should be in file: file1.out



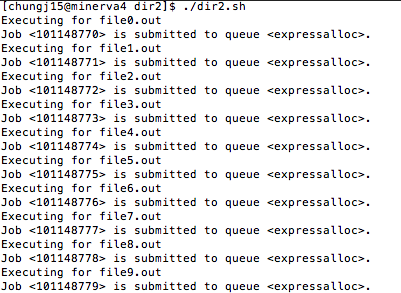




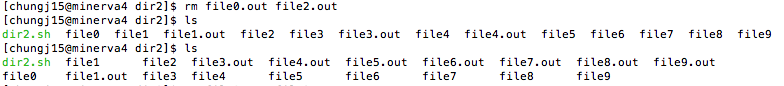
1. Write a script, or modify the answer to question 2, so that for each file it will submit a job to “cat” that file into file<x>.out only if the file<x>.out does not exist. The output file does not necessarily have to be STDOUT. Test it using dir2.

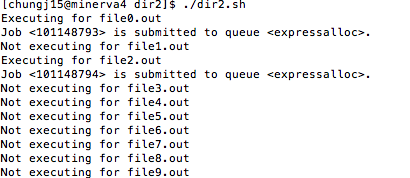
Example: file4 has a file4.out so you would not submit a job for file4; file1 does not have a file1.out so you would run a job for file1.





# remove 2 files to test





1. One of the things we’ve tried to do in this class is to make clear how to get help, and how to take advantage of libraries that might already implement the details of the method you are trying to use to get your work done. Some of the most important such libraries are those for linear algebra, which are found frequently in all of computational science, and they are written for various platforms. Find 3 such linear algebra libraries that implement matrix-matrix multiplication on different architectures (GPU, distributed memory, etc.) and describe at least one advantage and disadvantage that particular architecture has for the calculation. (Hint: speed, problem size, complexity, etc., might be things to think about as potential advantages).

* MAGMA (Matrix Algebra on GPU and Multicore Architectures) is the linear algebra library similar to LAPACK in functionality such as data storage and interface but it is different than LAPACK for heterogeneous (GPU and CPU) architectures. MAGMA achieves high performance and portability across multi-core architectures and hybrid systems. On the GPU, MAGMA can outperform corresponding packages by the power that each homogeneous components offers. The performance of MAGMA can exceed 10X the performance than a system with 48 modern CPU cores. The problem with MAGMA is due to the daunting computational landscape of hybrid processors. It’s hard to keep up with the pace of change: sometimes performance is not scaling up as the number of cores grows because more and more time is spent on slow data movement rather than fast arithmetic [1].
* ScaLAPACK (or Scalable LAPACK) library includes [LAPACK](https://en.wikipedia.org/wiki/LAPACK) routines and runs on [distributed memory](https://en.wikipedia.org/wiki/Distributed_memory) [MIMD](https://en.wikipedia.org/wiki/MIMD) [parallel computers](https://en.wikipedia.org/wiki/Parallel_computer).  ScaLAPACK can run on any machine where BLAS, LAPACK and the BLACS are available. Scientists have achieved the advantages of efficiency, scalability (as the problem size), reliability (including error bounds), portability (across all important parallel machines), flexibility (so users can construct new routines from well-designed parts), and ease of use (resemblance of interface to LAPACK) in ScaLAPACK [2]. However, each node's main memory needs to be loaded with the same software, for there is no common main memory that can be accessed by different processors or nodes [3]. In terms of functionality, ScaLAPACK misses some advanced algorithms – SVD and QR with pivoting least squares, generalized least squares, non-symmetric eigenvalue problems and D&C SVD.

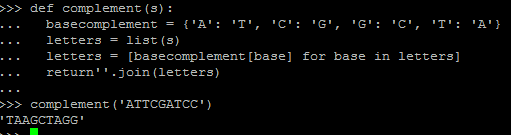
BLAS (Basic Linear Algebra Subroutines)-compatible libraries such as [AMD Core Math Library](https://en.wikipedia.org/wiki/AMD_Core_Math_Library) (ACML), [ATLAS](https://en.wikipedia.org/wiki/Automatically_Tuned_Linear_Algebra_Software), [Intel Math Kernel Library](https://en.wikipedia.org/wiki/Intel_Math_Kernel_Library) (MKL), and [OpenBLAS](https://en.wikipedia.org/wiki/OpenBLAS) function for a number of basic linear algebra operations on the hardware such as vector registers or [SIMD](https://en.wikipedia.org/wiki/SIMD) instructions. BLAS are efficient, portable, and widely available, and they can be useful for development of high quality linear algebra software [4]. Various storage formats can be obtained on four data types (single, double, complex, double complex). The major problem with BLAS is memory traffic, by which memory bandwidth and latency cannot match the high performance of floating point computations on the chip [5], unless it can be helped by inherent locality in space and time.

* [1] <http://icl.cs.utk.edu/news_pub/submissions/plasma-scidac09.pdf>
* [2] <http://www.netlib.org/scalapack/>
* [3] <https://www.packtpub.com/mapt/book/hardware_and_creative/9781783989447/4>
* [4] http://www.netlib.org/blas/
* [5] https://www8.cs.umu.se/kurser/5DV050/VT08/utdelat/F7.pdf

1. Write a python function that returns the compliment of a given DNA sequence. For example:

Given: ATTCGATCC

Output: TAAGCTAGG



1. A DNA string is reverse palindrome if it is equal to its reverse complement. For instance, the following portion of a DNA sequence in red is a reverse palindrome,

5’ . . . GAATTC . . . 3’ 🡨 (0)

. . . CTTAAG . . . 🡨 (1) complement

. . . GAATTC . . . 🡨 (2) reverse complement

The reverse complement (2) is obtained by flipping over the complement sequence (1). And the resulting sequence is exactly same as the top sequence (0).

Many restriction enzymes recognize specific palindromic sequences and cut them. Write a python program that finds the position and the length of every reverse palindrome in a given string. For example:

Given: TCAATGCATGCGGGTCTATATGCAT

Output:

4 6

5 4

6 6

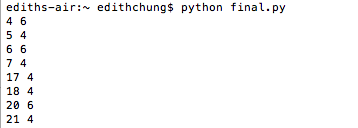
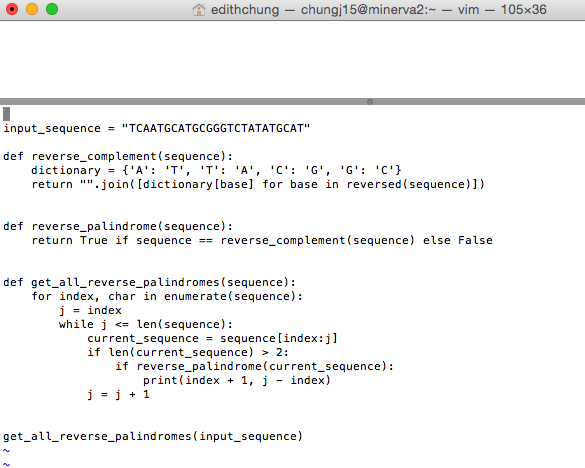
7 4

17 4

18 4

20 6

21 4



1. Write a python program that creates two random 2000 X 2000 floating point matrices with a uniform distribution between 0 and 1, and computes the product (matrix multiplication) of the two matrices using numpy.

