**Laborator 02**

**Setup infrastructură (Făcut în Laborator 01)**

* Verificați dacă folderul lab mai este mounted.
  + **mount | grep labs**
* Dacă nu este sau sunt probleme cu el:
  + **sudo pkill -9 sshfs**
  + **sudo umount /home/USER\_LOCAL/labs**
  + **sudo mount -a**
* Dacă vreți să verificați fișierele de pe server puteți folosi ssh sau WinSCP.
* Instalați [WinSCP](https://winscp.net/eng/index.php).
  + Când instalați, de la *User Interface Style* selectați *Commander*.
  + New Site
    - Host name: wiki.mta.ro
    - Port number: 30000
    - User name: cel de pe wiki.mta.ro
  + Advanced...
    - SSH >> Authentication >> Private Key File […] >> OK
  + Save >> Login
  + Stânga mergeți în directorul dorit - Dreapta folder-ul de pe server
  + Mergeți în folder-ul labs pe server.

[Tutorial llnl](https://computing.llnl.gov/tutorials/mpi/)  
[MPI The complete Reference](http://www.netlib.org/utk/papers/mpi-book/mpi-book.html)

**Exerciții**

**Pentru fiecare exercițiu se va scrie în fișierul REPORT.txt rezultatul rulărilor și răspunsul la întrebări.**

**Pentru fiecare exercițiu se vor face afișări care să evidențieze comportamentul (înainte și după apelul funcțiilor MPI).**

1. (**1\_helloWorld.c**) Citiți, compilați și rulați programul helloWorld. În Makefile aveți un exemplu de compilare. Rulare se va face folosind **mpirun -np 4 ./helloWorld**.
2. (**2\_3\_send.c**) Implementați un program MPI cu doua procese. Procesul 0 va trimite o valoare procesului doi. După primare acesta o va afișa. Aveți grijă să inițializați variabila doar pe procesul 0.
3. (**2\_3\_send.c**) Modificați programul anterior adăugând transmisia unui vector de 100 de elemente. Se va executa întreaga transmisie printr-un singur apel. Aveți grijă să inițializați vectorul doar pe procesul 0.
4. (**4\_5\_broadcast.c**) Implementați un program MPI cu 4 procese. Folosind brodcast un programul 2 trimite o valoare tuturor celelalte. Aveți grijă să inițializați variabila doar pe programul 2. După trimitere afișați variabila de pe toate procesele.
5. (**4\_5\_broadcast.c**) Modificați programul anterior în așa fel încât să adăugați transmisia unui vector de 100 de element. Se va executa întreaga transmisie printr-un singur apel. Aveți grijă să inițializați vectorul doar pe programul 2. După trimitere afișați vectorul de pe toate procesele.
6. (**6\_scatterGather.c**) Implementați un program MPI cu 4 procese. Procesul 0 inițializează vectorul de 100 de elemente după regula **v[i]=i**. Vectorul este împărțit tuturor proceselor. Fiecare din cele 4 procese adună valoarea 42 elementelor din vector de care este responsabil (25 de elemente fiecare). După adunări, vectorul va fi colectat pe procesul 0 și afișat complet.
7. (**7\_circle.c**) Implementați programul MPI cu **N** procese. Procesul 0 trimite procesului următor valoarea **1**. Toate celelalte procese primesc valoarea de la procesul dinaintea lor, adaugă **2** la ea și trimit valoarea mai departe procesului următor. Ultimul proces, după adunare, trimite valoarea procesului 0, formând un cerc. La fiecare send, recv, și adunare se vor face afișări.
8. (**8\_anySource.c**) Scrieți un program MPI cu 4 procese.
   * Primele 3 procese trimit o valoare ultimului.
   * Ultimul proces primește cele trei valori cu **MPI\_ANY\_SOURCE**.
   * Se printează de pe procesul **3** valoarea și sursa din **MPI\_Status**.
9. (**9\_anyTag.c**) Scrieți un program MPI cu 2 procese.
   * Procesul **0** trimite 3 valori procesului **1**, fiecare cu alt tag.
   * Procesul **1** va primi valorile folosind **MPI\_ANY\_TAG**.
   * Se printează valorile de pe procesul **1** și tag-ul din **MPI\_Status**.

**Exercițiile de la 1 la 9** sunt **obligatorii**. Conceptele explorate sunt esențiale pentru obținerea notei **minime** de promovare.

**Vă recomandăm, pentru a crește șansele de a obține o notă cât mai mare să explorați și următoarele exerciții:**

1. (**10\_findNum.c**) Implementați un joc de descoperire a unui număr.
   * Considerăm procesul cu rank 1 ca fiind server.
   * Serverul alege un număr random între 0 și 100.
   * Când primește un mesaj cu un număr serverul răspunde la acesta precizând dacă numărul primit este mai mare sau mai mic decât cel ales.
   * Considerăm procesul cu rank 0 ca fiind client.
   * Clientul citește de la tastatură un număr, și îl trimite serverului.
   * Tot clientul afișează răspunsul serverlui pe ecran.
   * Va trebui să aveți un mesaj special pentru a opri programul distribuit.
2. (**11\_findNumAuto.c**) Se modifică programul anterior pentru a schimba clientul cu unul care va căuta singur numărul ales de server.
   * Pentru a găsi numărul cât mai repede, clientul va face căutare binară.

**Common problems:**

Dacă aveți problema următoare când rulați cu mpirun:

Text

Description automatically generated

Pentru a rezolva rulați ca root comanda:

**echo 0 > /proc/sys/kernel/yama/ptrace\_scope**

Dacă aveți o problemă de genul când rulați cu mpirun:

Text

Description automatically generated

Adăugați comenzii mpirun parametrul **--oversubscribe**