



## Laborator 01

### Setup infrastructură

- Instalați Windows Subsystems for Linux.
  - Control Panel >> **Windows Features** >> Selectați **Windows Subsystems for Linux** și **Virtual Machine Platform** >> **OK**
- În cmd dați comanda `wsl --set-default-version 2`
- Este posibil să fie nevoie să activați din BIOS virtualizare.
- Instalați Ubuntu 20.04.
  - Microsoft Store >> Search Ubuntu >> Ubuntu 20.04 >> Install >> Launch
- Descărcați cheia de pe [wiki.mta.ro](http://wiki.mta.ro).
- Instalați [Putty](#) și testați conexiunea ssh.
  - Host Name: [USER\\_WIKI@wiki.mta.ro](mailto:USER_WIKI@wiki.mta.ro)
  - Port 30000
  - Connections >> SSH >> Auth >> Browse... pentru a pune cheia (.ppk).
  - **Întorceți-vă la meniul inițial și dați SAVE.**
- **Pe UBUNTU local:** instalați compilator, make și sshfs pe Linux.
  - `sudo apt-get update`
  - `sudo apt-get install gcc make gdb sshfs ssh`
- Testați conexiunea ssh.
  - `ssh -i CHEIE.key -p 30000 USER_WIKI@wiki.mta.ro`
- Testați montarea directorului de laboratoare folosind sshfs
  - Mare atenție, nu puteți da copy paste la comenzi, trebuie să le modificați în așa fel încât să se potrivească sistemului vostru.
  - Copiați cheia (.key) de la asistent în /home/USERNAME\_UL\_VOSTRU/
  - Setări drepturi restrânse cheii: `chmod 400 /home/USER_LOCAL/CHEIE.key`
  - Creați un director `mkdir /home/USER_LOCAL/labs`
  - `sshfs -o IdentityFile=/home/USER_LOCAL/CHEIE.key -p 30000 USER_WIKI@wiki.mta.ro: /home/USER_LOCAL/labs`
  - Ar trebui să puteți intra în folder și să lucrați.
  - Dacă nu a mers dați comanda `mount` (ultima linie).
  - Dacă în mount apare directorul vostru `sudo umount /home/USER_LOCAL/labs`
- Montare permanentă:
  - În fișierul `/etc/fstab` adăugați linia:
  - `sshfs#USER_WIKI@wiki.mta.ro: /home/USER_LOCAL/labs fuse defaults,allow_other,reconnect,IdentityFile=/home/USER_LOCAL/KEY.key,port=30000 0 0`
  - pentru a porni pentru prima oară montarea `sudo mount -a`
- Instalați [Visual Studio Code](#).
- Setări Visual Studio Code să folosească WSL (Windows Subsystems for Linux).
  - Stânga jos buton verde două săgeți
  - Remote-WSL: New Window
    - Dacă aveți mai multe distribuții instalate e bine să apăsați Remote-WSL: New Window using Distro... și apoi să o selectați pe cea cu Ubuntu 20.04
  - Open folder... și alegeți folderul `/home/USER_LOCAL/labs/lab01`
  - **Trebuie să apară în Visual Studio subfolderul .vscode**
- Instalați extensii Visual Studio Code:
  - Remote-WSL – autor Microsoft
  - C/C++ (IntelliSense) – autor Microsoft


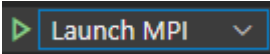


- Instalați MPI pe Linux.
  - `sudo apt-get install libopenmpi-dev openmpi-bin`
  - `sudo apt-get install openmpi-doc openmpi-common`

[Tutorial IInI](#)  
[MPI The complete Reference](#)

### Exerciții

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

1. **(1\_helloWorld.c)** Compilați și rulați codul.
    - Rulați din Visual Studio Code apăsând  apoi 
    - Din terminal:
      - Compiare: `mpicc -o helloWorld helloWorld.c`
      - Rulare: `mpirun -n NUM_PROCESSES ./helloWorld`
  2. **(2\_numCores.c)** Aflați numărul de core-uri ale procesorului folosit, din linia de comandă și din codul C.
    - Căutați pe Google cum se afișează numărul de core-uri din CLI pe Linux
    - Căutați pe Google “sysconf() number of cores”
  3. Rulați programul de la 1, cu 3 procese din VS Code și din linia de comandă.
    - Din VS Code se poate modifica din `.vscode/launch.json`, parametrul `args`.
  4. Rulați programul de la 1, cu 20 procese.
    - De ce funcționează un program cu mai multe procese decât core-uri?
  5. **(3\_print100.c)** Modificați codul.
    - Programul se va rula cu 2 procese.
    - Mesajul “Hello World from x/y at i” va fi afișat de 100 de ori.
    - În loc de x va fi afișat id-ul (rank) procesului.
    - În loc de y va fi afișat numărul total de procese (nprocesses).
    - **În loc de i va fi afișat identificatorul iterației.**
    - Cum arată afișarea? Explicați.
    - Dacă nu se comportă cum vă așteptați măriți numărul de procese/iterații.
  6. **(4\_twoDifferentProcesses.c)** Modificați codul.
    - Programul se va porni cu 2 procese.
    - Unul din procese va apela funcția **printHelloWorld()**.
    - Al doilea proces va apela funcția **printSomethingElse()**.
    - Voi va trebui să implementați cele două funcții.
  7. **(5\_firstAndLast.c)** Modificați codul.
    - Programul va fi pornit cu 5 procese.
    - Toate afișează mesajul de “Hello World”.
    - Doar primul proces afișează “Mesaj de la primul”, alături de rank-ul său.
    - Doar ultimul proces afișează “Mesaj de la ultimul”, alături de rank-ul său.
-



**Exercițiile de la 1 la 7 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:**

8. **(6\_variables.c)** Modificați codul.
- Programul va fi pornit cu 10 procese.
  - Se declară și inițializează pe toate procesele variabila A cu 2.
  - Se declară pe toate procesele variabila B.
    - Toate procesele inițializează B cu 0.
    - Primul proces modifică B la 100.
    - Ultimul proces modifică B la 1000.
  - Se afișează de pe toate procesele alături de mesajul de "Hello World" valoarea  $A^{rank} + B$ .

### Hints:

Dacă aveți problema următoare când rulați cu mpirun:

```
-----  
WARNING: Linux kernel CMA support was requested via the  
btl_vader_single_copy_mechanism MCA variable, but CMA support is  
not available due to restrictive ptrace settings.  
  
The vader shared memory BTL will fall back on another single-copy  
mechanism if one is available. This may result in lower performance.
```

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:



```
There are not enough slots available in the system to satisfy the 100 slots that were requested by the application:
```

```
./helloWorld
```

```
Either request fewer slots for your application, or make more slots available for use.
```

```
A "slot" is the Open MPI term for an allocatable unit where we can launch a process. The number of slots available are defined by the environment in which Open MPI processes are run:
```

Adăugați comenzii mpirun parametrul **--oversubscribe**

Este primul an în care folosim sshfs. Dacă se blochează încercați:

- `killall -9 sshfs`
- `umount /home/USER_LOCAL/labs`
- `mount -a`

În cazul în care acesta face în continuare probleme putem folosi winscp pe Windows direct.

- În momentul în care alegeți folderul în care să lucrați din vscode din modul remote WSL **scrieți /mnt/ în loc de /root** . Selectați partiția și acum sunteți prezentat cu lista de directoare Windows. Folosiți directorul în care doriți să lucrați.
- Instalați [WinSCP](#).
  - 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.
  - La începutul laboratorului copiați de pe server pe local.
    - **Atenție** copiați tot directorul labs (folderul 01 conține un folder .vscode care este invizibil și nu va fi copiat dacă copiați fișierele din director).
  - **La sfârșitul laboratorului copiați de pe local pe server.**