Wprowadzenie do infrastruktury obliczeniowej EuroHPC

Michał Hermanowicz, Jan Miśkiewicz

Zespół Oprogramowania Aplikacyjnego i Wsparcia Użytkowników, ICM, Uniwersytet Warszawski

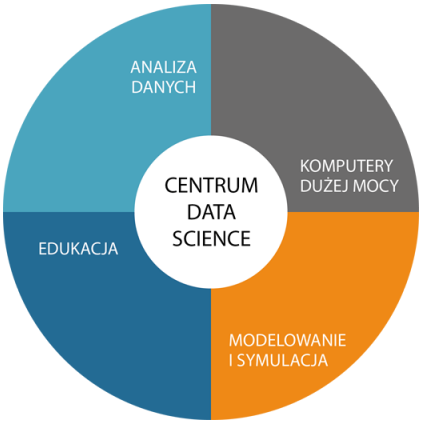
14 marca 2023

Rzeczpospolita Polska

Unia Europejska
Europejski Fundusz Rozwoju Regionalnego

Spis rzeczy

Wprowadzenie



Naszą misją jest rozumieć dane i dostarczać innowacyjne rozwiązania organizacjom i instytucjom dzięki wykorzystaniu specjalistycznych kompetencji z zakresu data science.

- Infrastruktura obliczeniowa i analityczna,
- współpraca projektowej: wsparcie techniczne i programistyczne,
- konsultacje i szkolenia.

OWU: Zespół Oprogramowania Aplikacyjnego i Wsparcia Użytkowników

Do podstawowych obowiązków OWU należą:

- Wsparcie dla użytkowników HPC,
- wdrażanie, testowanie i monitorowanie oprogramowania aplikacyjnego,
- angażowanie się w konferencje tematyczne i działania badawcze,
- tworzenie i utrzymywanie dokumentacji oprogramowania,
- organizowanie samouczków i sesji szkoleniowych.

Projekt EuroHPC: https://eurohpc-ju.europa.eu

EuroHPC JU: European European High Performance Computing Joint Undertaking

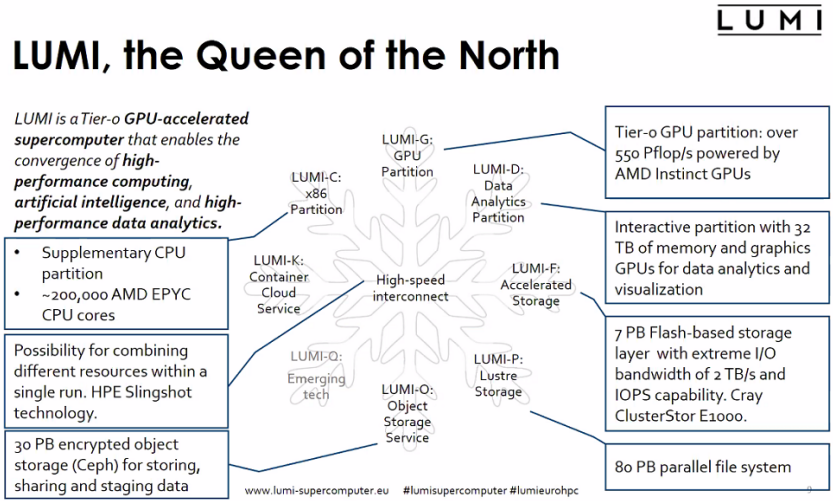
- Rozwój i utrzymanie zlokalizowanych w UE zasobów HPC i infrastruktury towarzyszącej,
- zapewnienie łańcucha dostaw niezbędnych komponentów, technologii i wiedzy,
- rozszerzenie wykorzystania infrastruktury superkomputerowej w sektorach publicznym i prywatnym oraz wsparcie rozwoju kompetencji HPC.

Superkomputery:

LUMI, LEONARDO, MERENOSTRUM 5, VEGA, MELUXINA, KAROLINA, DISCOVERER, DEUCALION

Projekt EuroHPC: https://eurohpc-ju.europa.eu

Projekt EuroHPC PL: https://eurohpc.pl


LUMI
LUMI, the Queen of the North
LUMI is a Tier-0 GPU-accelerated supercomputer that enables the convergence of high-performance computing, artificial intelligence, and high-performance data analytics.
• Supplementary CPU partition
• ~200,000 AMD EPYC CPU cores
Possibility for combining different resources within a single run. HPE Slingshot technology.
30 PB encrypted object storage (Ceph) for storing, sharing and staging data
LUMI-G: GPU Partition
LUMI-C: x86 Partition
LUMI-D: Data Analytics Partition
LUMI-K: Container Cloud Service
High-speed interconnect
LUMI-F: Accelerated Storage
LUMI-Q: Emerging tech
LUMI-P: Lustre Storage
LUMI-O: Object Storage Service
Tier-0 GPU partition: over 550 Pflop/s powered by AMD Instinct GPUs
Interactive partition with 32 TB of memory and graphics GPUs for data analytics and visualization
7 PB Flash-based storage layer with extreme I/O bandwidth of 2 TB/s and IOPS capability. Cray ClusterStor E1000.
80 PB parallel file system
www.lumi-supercomputer.eu #lumisupercomputer #lumieurohpc

Superkomputer LUMI (Large Unified Modern Infrastructure)

Konfiguracja:

- LUMI-C
  - 1536 węzłów (2×64-core AMD EPYC 7763),
  - 256, 512, 1024 GB pamięci operacyjnej,
- LUMI-G
  - 2560 węzłów (1×AMD EPYC 7A53) + 4 AMD MI250x GPU,
  - (512 GB + 4x128 GB RAM).

Superkomputer LUMI (Large Unified Modern Infrastructure)

Konfiguracja:

- LUMI-C
  - 1536 węzłów (2×64-core AMD EPYC 7763),
  - 256, 512, 1024 GB pamięci operacyjnej,
- LUMI-G
  - 2560 węzłów (1×AMD EPYC 7A53) + 4 AMD MI250x GPU,
  - (512 GB + 4x128 GB RAM).

- Dostęp do LUMI koordynuje ACK Cyfronet AGH (PL-Grid),
- granty obliczeniowe (konkursy) i testowe (miesięczne; w trybie ciągłym),
- w dostępie testowym: 10 tys. godzin CPU. (niezbędne sprawozdanie),
- granty pilotazowe dla naukowców z Polski.

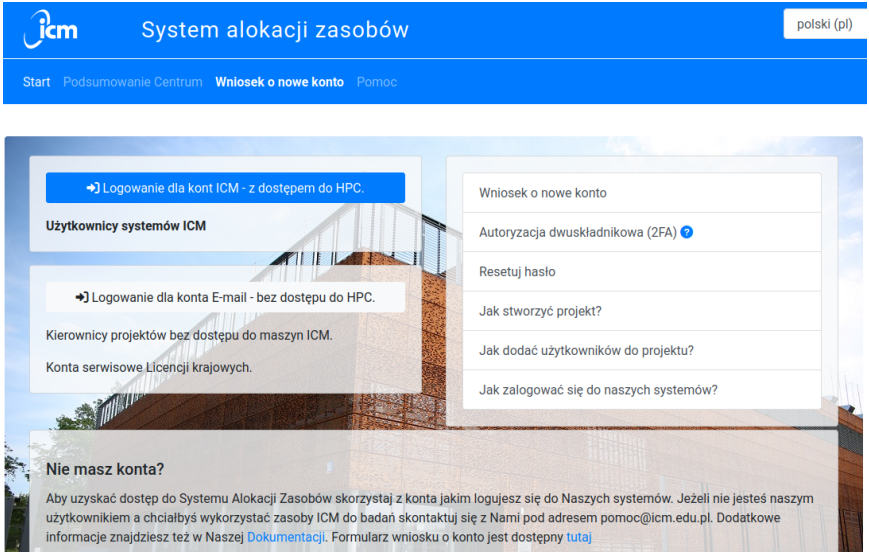
EuroHPC i Narodowe Centra Kompetencji (EuroCC 2)

EuroCC 2 – NCC:

- Polska tworzy jeden z (ponad 30) **Narodowych Centrów Kompetencji** w ramach projektu EuroHPC,
- NCC operują na poziomie regionalnym/krajowym (dostęp do zasobów EuroHPC),
- organizują szkolenia, utrzymują kontakt z sektorem publicznym i prywatnym, buduja bazę kompetencji,
- wzmacniają technologiczną autonomię UE w zakresie HPC.

Dostęp i użytkowanie

System Alokacji Zasobów → https://granty.icm.edu.pl



System Alokacji Zasobów → https://granty.icm.edu.pl

Procedura uzyskiwania dostępu do zasobów ICM

- Konto użytkownika
  - Projekt
    - Alokacja (zasób obliczeniowy)

- Wnioskowanie o projekt wymaga statusu PI (kierownik projektu)
  → *Wystąp o konto rozszerzone*,
- Sprawozdanie z obliczeń – składane raz w roku.

System Alokacji Zasobów → https://granty.icm.edu.pl

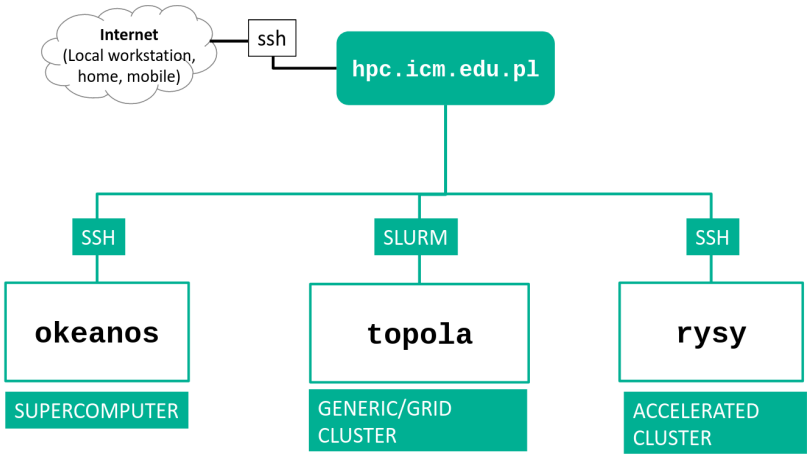
Procedura uzyskiwania dostępu do zasobów ICM

- Konto użytkownika
  - Projekt
    - Alokacja (zasób obliczeniowy)

- Wnioskowanie o projekt wymaga statusu PI (kierownik projektu)
  → *Wystąp o konto rozszerzone*,
- Sprawozdanie z obliczeń – składane raz w roku.

Wnioski o alokacje są rozpatrywane w trybie ciągłym i podlegają recenzji.

Infrastruktura ICM



Maszyny obliczeniowe – Topola, Okeanos

System: **Okeanos (Cray XC40)**
Architecture: Intel Xeon E5-2690 v3 2.6 GHz
CPU cores: 26 016
Single node parameters:

- Cores: 24 (2x12)
- RAM: 128 GB

System: **Topola (Huawei E9000 Cluster)**
Architecture: Intel Xeon E5-2697 v3 2.1 GHz
CPU cores: 6244
Single nodeparameters:

- Cores: 28
- RAM: 64/128 GB

Maszyny obliczeniowe – Rysy (GPU, VPU)

GPU

System: **GPU Cluster**
Arch: Intel Skylake / NV Volta
Compute nodes: 7
Single node parameters:

- Cores: 36
- RAM: 380 GB
- GPU: 4

VE

System: **NEC SX-Aurora Tsubasa**
Architecture:

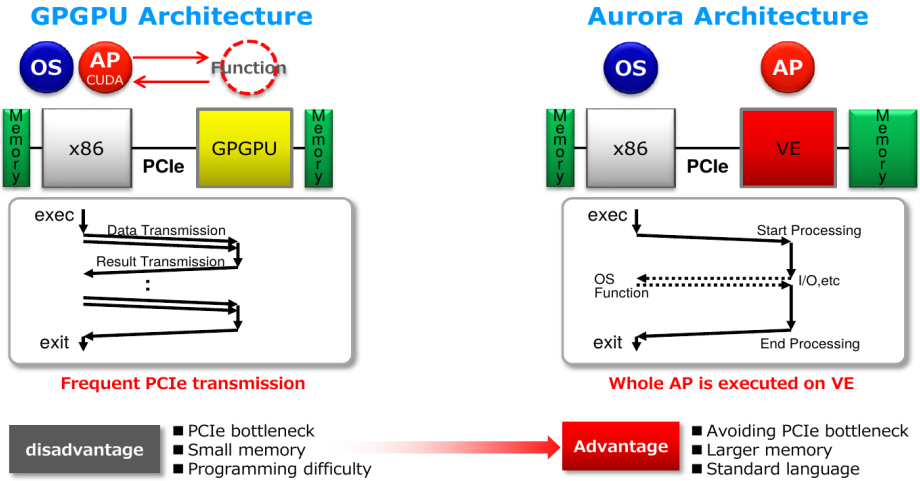
- VE: SX-Aurora Tsubasa
- VH: Intel Xeon Gold 6126

Compute nodes: 1
Single node parameters:

- Cores: 8 x 8 (VE) + 2 x 12 (VH)
- RAM: 8 x 48 GB (VE) + 192 GB (VH)

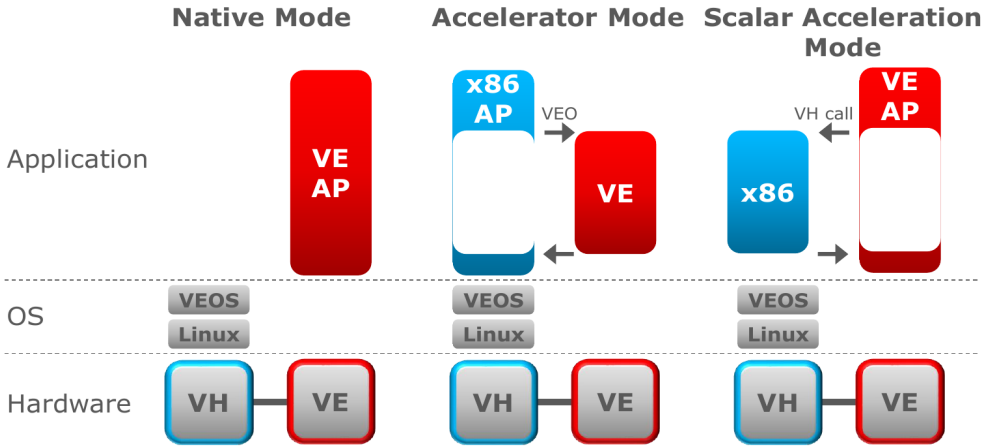
NEC
Orchestrating a brighter world

Maszyny obliczeniowe – Rysy (GPU, VPU)



Materiały: (C) NEC Corporation

Maszyny obliczeniowe – Rysy (GPU, VPU)



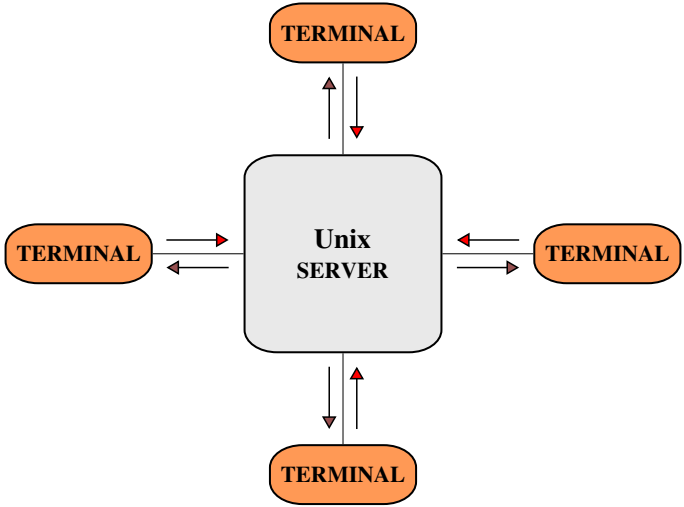
Materiały: (C) NEC Corporation

Ekosystem oprogramowania

Ken Thompson i Dennis Ritchie (AT&T Bell Labs)

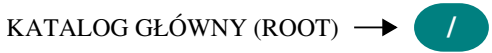
Zdjęcie: **Peter Hamer** [CC BY-SA 2.0 (http://creativecommons.org/licenses/by-sa/2.0)], via Wikimedia Commons

System operacyjny Unix: środowisko wieloużytkowe

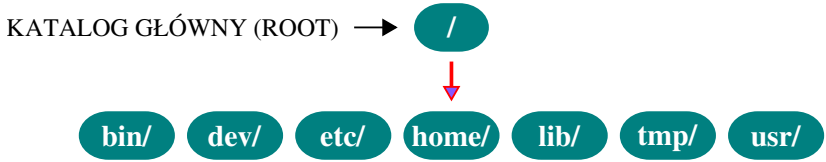


System operacyjny Unix: terminal **VT100**

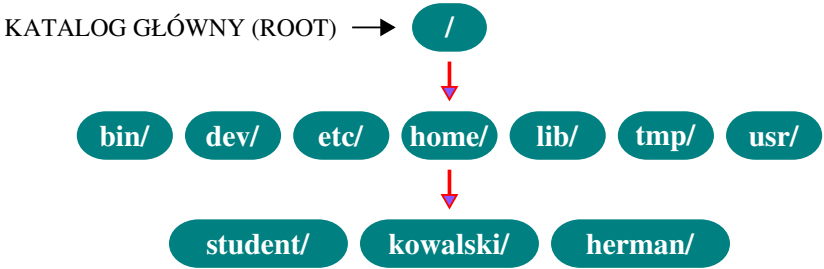
Zdjęcie: **Jason Scott** [CC BY 2.0 (http://creativecommons.org/licenses/by/2.0)], via Wikimedia Commons


KATALOG GŁÓWNY (ROOT)
/

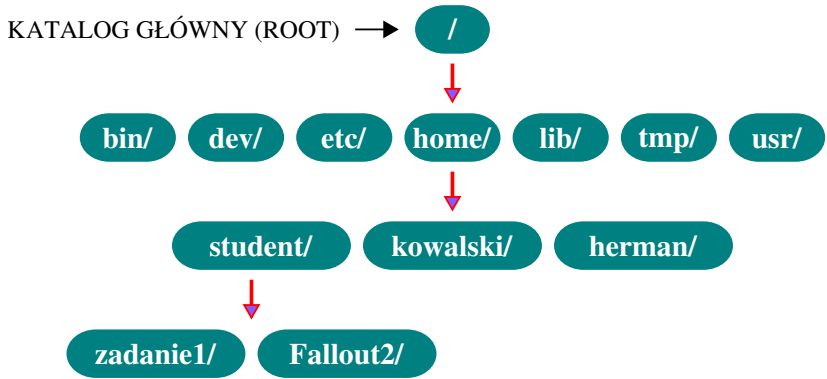
System operacyjny GNU/Linux: **system plików**



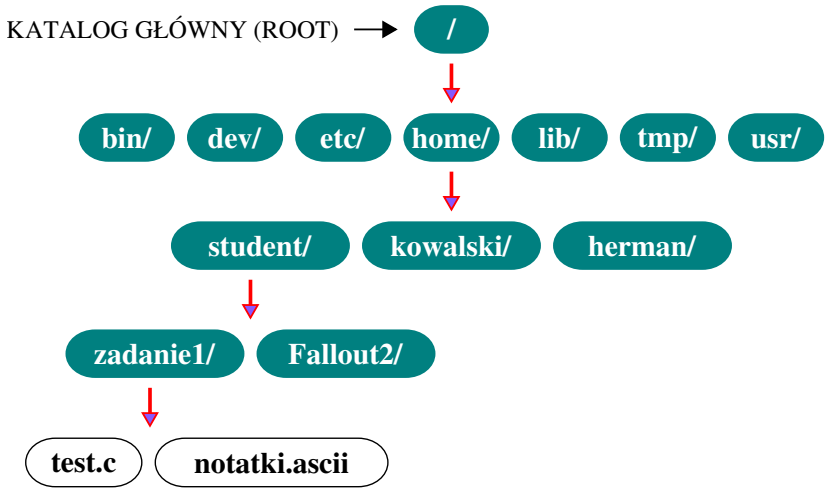
System operacyjny GNU/Linux: **system plików**



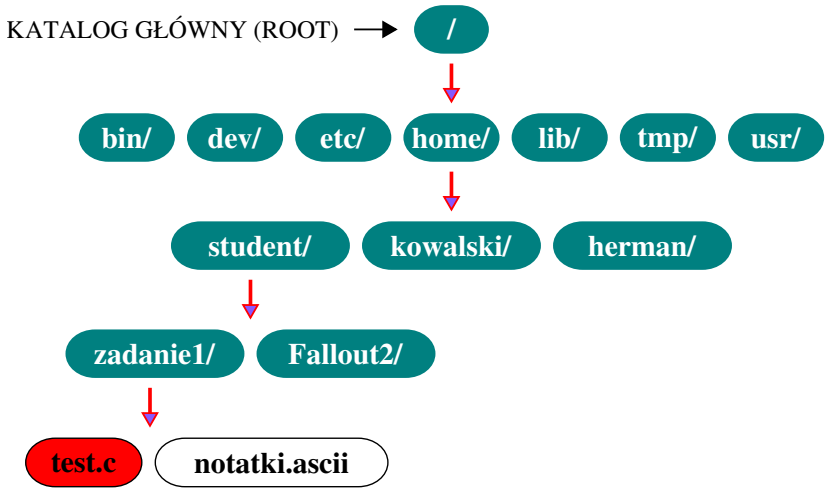
System operacyjny GNU/Linux: **system plików**



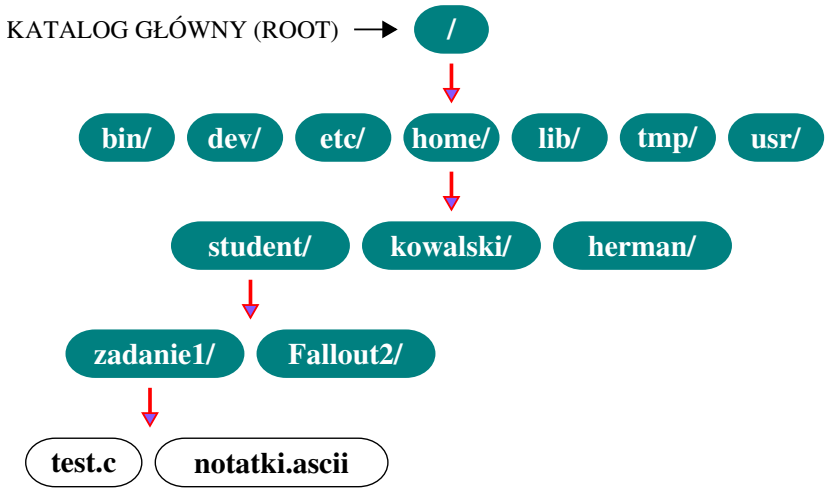
System operacyjny GNU/Linux: **system plików**



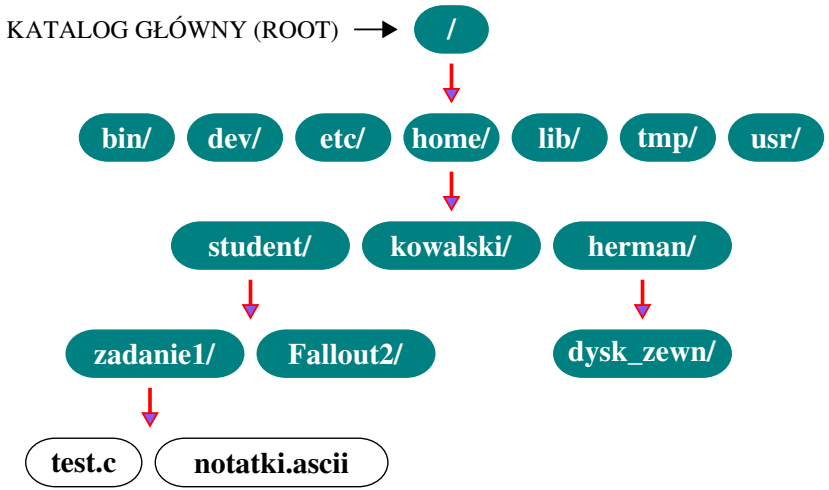
System operacyjny GNU/Linux: **system plików**



System operacyjny GNU/Linux: **system plików**



System operacyjny GNU/Linux: **system plików**



Kopiowanie/przenoszenie plików i kopie zapasowe

Kopiowanie plików:

- `cp /sciezka/dostepu/plik /sciezka/dostepu/plik2`
- `cp plik1 plik2 plik3 plik4 plik5 /katalog/docelowy`
- `cp -R katalog_zrodlowy katalog_docelowy`

Kopiowanie/przenoszenie plików i kopie zapasowe

Kopiowanie plików:

- `cp /sciezka/dostepu/plik /sciezka/dostepu/plik2`
- `cp plik1 plik2 plik3 plik4 plik5 /katalog/docelowy`
- `cp -R katalog_zrodlowy katalog_docelowy`

Przenoszenie/zmiana nazwy pliku:

- `mv plik1 plik2`

Kopiowanie/przenoszenie plików i kopie zapasowe

Kopiowanie plików:

- `cp /sciezka/dostepu/plik /sciezka/dostepu/plik2`
- `cp plik1 plik2 plik3 plik4 plik5 /katalog/docelowy`
- `cp -R katalog_zrodlowy katalog_docelowy`

Przenoszenie/zmiana nazwy pliku:

- `mv plik1 plik2`

Wykonywanie kopii zapasowej (*backup*):

polecenie `rsync` umożliwia lokalne i zdalne kopiowanie plików (`man rsync`).

Kopiowanie/przenoszenie plików i kopie zapasowe

Kopiowanie plików:

- `cp /sciezka/dostepu/plik /sciezka/dostepu/plik2`
- `cp plik1 plik2 plik3 plik4 plik5 /katalog/docelowy`
- `cp -R katalog_zrodlowy katalog_docelowy`

Przenoszenie/zmiana nazwy pliku:

- `mv plik1 plik2`

Wykonywanie kopii zapasowej (*backup*):

polecenie `rsync` umożliwia lokalne i zdalne kopiowanie plików (`man rsync`).

```
rsync -avuh --progress /home/herman /mnt/drugi_dysk/
```

Kopiowanie/przenoszenie plików i kopie zapasowe

Kopiowanie plików:

- `cp /sciezka/dostepu/plik /sciezka/dostepu/plik2`
- `cp plik1 plik2 plik3 plik4 plik5 /katalog/docelowy`
- `cp -R katalog_zrodlowy katalog_docelowy`

Przenoszenie/zmiana nazwy pliku:

- `mv plik1 plik2`

Wykonywanie kopii zapasowej (*backup*):

polecenie `rsync` umożliwia lokalne i zdalne kopiowanie plików (`man rsync`).

```
rsync -avuh --progress /home/herman /mnt/drugi_dysk/
rsync -avuh --progress ./herman herman@serwer.pl:/home/
```

Bezwzględna ścieżka dostępu:

```
/home/student/zadanie1/test.c
```

Bezwzględna ścieżka dostępu:

```
/home/student/zadanie1/test.c
```

Względna ścieżka dostępu:

```
zadanie1/test.c
```

System operacyjny GNU/Linux: **system plików**

Bezwzględna ścieżka dostępu:

`/home/student/zadanie1/test.c`

Względna ścieżka dostępu:

`zadanie1/test.c`

Ważne katalogi:

- `/bin` – pliki wykonywalne (***bin****ary*), programy użytkowe i inne,
- `/dev` – pliki urządzeń (dyski, drukarki i inne),
- `/etc` – pliki konfiguracyjne systemu,
- `/home` – katalogi domowe użytkowników,
- `/lib` – biblioteki systemowe,
- `/tmp` – pliki tymczasowe,
- `/usr` – oprogramowanie użytkownika wraz z bibliotekami, dok. i in.

W nazwach plików i katalogów:

- **NIE** używamy SPACJI*,
- **NIE** używamy znaków diakrytycznych*.

*Choć jest to technicznie możliwe, to znacznie komplikuje wiele czynności i w efekcie utrudnia korzystanie z systemu, dlatego taką zasadę warto przyjąć jako dobrą praktykę.

W nazwach plików i katalogów:

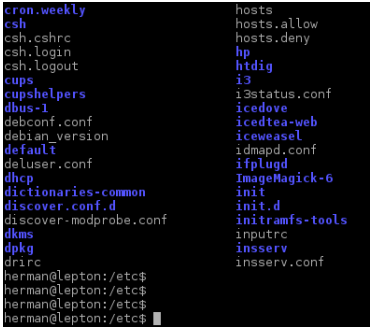
- **NIE** używamy SPACJI*,
- **NIE** używamy znaków diakrytycznych*.

*Choć jest to technicznie możliwe, to znacznie komplikuje wiele czynności i w efekcie utrudnia korzystanie z systemu, dlatego taką zasadę warto przyjąć jako dobrą praktykę.

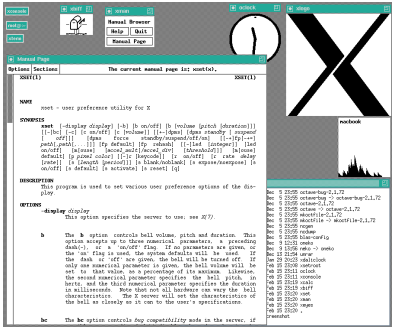
Domyślnie użytkownik ma uprawnienia do zapisu w katalogach:

- `/lu/topola/home/$USER`
- `/lu/tetyda/home/$USER`

System operacyjny GNU/Linux: **interfejs użytkownika**



CLI
(Command Line Interface)



GUI
(Graphical User Interface)

Zrzut ekranu z prawej: Liberal Classic [MIT (http://opensource.org/licenses/mit-license.php)], via Wikimedia Commons

GNU Bash (**B**ourne **A**gain **SH**ell):

interpreter języka poleceń – **powłoka** (*shell*) systemu GNU/Linux[1].

[1] *The GNU Bash Reference Manual*, v. 4.3 (http://www.gnu.org).

System operacyjny GNU/Linux: CLI – **Bash**

GNU Bash (**B**ourne **A**gain **SH**ell):

interpreter języka poleceń – **powłoka** (*shell*) systemu GNU/Linux[1].

[1] *The GNU Bash Reference Manual*, v. 4.3 (http://www.gnu.org).

Ponadto:

- jest domyślną powłoką systemu GNU/Linux – jego **CLI** (Command Line Interface),
- umożliwia pracę interaktywną (wprowadzanie poleceń), a także wsadową (wykonywanie skryptów).

Inne powłoki: `sh`, `csh`, `tcsh`, `ksh`, `zsh`.

System operacyjny GNU/Linux: **CLI**

```
Debian GNU/Linux wftlab-180 tty1
wftlab-180 login:
```

System operacyjny GNU/Linux: **CLI**

```
Debian GNU/Linux wftlab-180 tty1
wftlab-180 login: student
```

System operacyjny GNU/Linux: **CLI**

```
Debian GNU/Linux wftlab-180 tty1

wftlab-180 login: student
Password:
```

System operacyjny GNU/Linux: **CLI**

```
Debian GNU/Linux wftlab-180 tty1

wftlab-180 login: student
Password:

student@wftlab-180:~$
```

System operacyjny GNU/Linux: **CLI**

```
Debian GNU/Linux wftlab-180 tty1

wftlab-180 login: student
Password:

student@wftlab-180:~$
student@wftlab-180:~$
```

System operacyjny GNU/Linux: **CLI**

```
Debian GNU/Linux wftlab-180 tty1

wftlab-180 login: student
Password:

student@wftlab-180:~$
student@wftlab-180:~$
student@wftlab-180:~$
```

System operacyjny GNU/Linux: **CLI**

```
Debian GNU/Linux wftlab-180 tty1

wftlab-180 login: student
Password:

student@wftlab-180:~$
student@wftlab-180:~$
student@wftlab-180:~$
student@wftlab-180:~$
```

System operacyjny GNU/Linux: **CLI**

```
Debian GNU/Linux wftlab-180 tty1

wftlab-180 login: student
Password:

student@wftlab-180:~$
student@wftlab-180:~$
student@wftlab-180:~$
student@wftlab-180:~$ pwd
```

System operacyjny GNU/Linux: **CLI**

```
Debian GNU/Linux wftlab-180 tty1

wftlab-180 login: student
Password:

student@wftlab-180:~$
student@wftlab-180:~$
student@wftlab-180:~$
student@wftlab-180:~$ pwd
/home/student
student@wftlab-180:~$
```

System operacyjny GNU/Linux: **CLI**

```
Debian GNU/Linux wftlab-180 tty1

wftlab-180 login: student
Password:

student@wftlab-180:~$
student@wftlab-180:~$
student@wftlab-180:~$
student@wftlab-180:~$ pwd
/home/student
student@wftlab-180:~$ ls
```

System operacyjny GNU/Linux: **CLI**

```
Debian GNU/Linux wftlab-180 tty1

wftlab-180 login: student
Password:

student@wftlab-180:~$
student@wftlab-180:~$
student@wftlab-180:~$
student@wftlab-180:~$ pwd
/home/student
student@wftlab-180:~$ ls
dokumenty gry na_zalke test123 zadanie1
student@wftlab-180:~$
```

System operacyjny GNU/Linux: **CLI**

```
Debian GNU/Linux wftlab-180 tty1

wftlab-180 login: student
Password:

student@wftlab-180:~$
student@wftlab-180:~$
student@wftlab-180:~$
student@wftlab-180:~$ pwd
/home/student
student@wftlab-180:~$ ls
dokumenty gry na_zalke test123 zadanie1
student@wftlab-180:~$
```

Uruchamianie programu – składnia:

```
program [OPCJA] [ARGUMENT]
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~$
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~$ ls
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~$ ls zadanie1
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~$ ls zadanie1
main.c notatki.ascii
student@wftlab-180:~$
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~$ ls zadanie1
main.c notatki.ascii
student@wftlab-180:~$ ls -l zadanie1
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~$ ls zadanie1
main.c notatki.ascii
student@wftlab-180:~$ ls -l zadanie1
razem 0
-rw-r--r-- 1 student student 0 wrz 1 23:20 main.c
-rw-r--r-- 1 student student 0 wrz 1 23:20 notatki.ascii
student@wftlab-180:~$
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~$ ls zadanie1
main.c notatki.ascii
student@wftlab-180:~$ ls -l zadanie1
razem 0
-rw-r--r-- 1 student student 0 wrz 1 23:20 main.c
-rw-r--r-- 1 student student 0 wrz 1 23:20 notatki.ascii
student@wftlab-180:~$ ls -a zadanie1
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~$ ls zadanie1
main.c notatki.ascii
student@wftlab-180:~$ ls -l zadanie1
razem 0
-rw-r--r-- 1 student student 0 wrz 1 23:20 main.c
-rw-r--r-- 1 student student 0 wrz 1 23:20 notatki.ascii
student@wftlab-180:~$ ls -a zadanie1
. .. .x main.c notatki.ascii
student@wftlab-180:~$
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~$ ls zadanie1
main.c notatki.ascii
student@wftlab-180:~$ ls -l zadanie1
razem 0
-rw-r--r-- 1 student student 0 wrz 1 23:20 main.c
-rw-r--r-- 1 student student 0 wrz 1 23:20 notatki.ascii
student@wftlab-180:~$ ls -a zadanie1
. .. .x main.c notatki.ascii
student@wftlab-180:~$
```

`.x`	plik/katalog **ukryty**
`..`	katalog **nad**rzędny
`.`	katalog **bieżący**
`~`	katalog **domowy**

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~$
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~$ cd
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~$ cd zadanie1
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~$ cd zadanie1
student@wftlab-180:~/zadanie1$
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~$ cd zadanie1
student@wftlab-180:~/zadanie1$ ls -l main.c
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~$ cd zadanie1
student@wftlab-180:~/zadanie1$ ls -l main.c
-rw-r--r-- 1 student student 0 wrz 1 23:20 main.c
student@wftlab-180:~/zadanie1$
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~$ cd zadanie1
student@wftlab-180:~/zadanie1$ ls -l main.c
-rw-r--r-- 1 student student 0 wrz 1 23:20 main.c
student@wftlab-180:~/zadanie1$
```

Użytkownicy należą do grup. Prawa dostępu mogą dotyczyć indywidualnych użytkowników, całych grup lub wszystkich razem.

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~$ cd zadanie1
student@wftlab-180:~/zadanie1$ ls -l main.c
-rw-r--r-- 1 student student 0 wrz 1 23:20 main.c
student@wftlab-180:~/zadanie1$
```

Użytkownicy należą do grup. Prawa dostępu mogą dotyczyć indywidualnych użytkowników, całych grup lub wszystkich razem.

Format praw dostępu – 4 bloki

```
-  rwx  rwx  rwx
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~$ cd zadanie1
student@wftlab-180:~/zadanie1$ ls -l main.c
-rw-r--r-- 1 student student 0 wrz 1 23:20 main.c
student@wftlab-180:~/zadanie1$
```

Użytkownicy należą do grup. Prawa dostępu mogą dotyczyć indywidualnych użytkowników, całych grup lub wszystkich razem.

Format praw dostępu – 4 bloki

`-  rwx  rwx  rwx`

- 1. blok (`-`): znak specjalny (plik/katalog),
- 2. blok (`rwx`): uprawnienia właściciela pliku,
- 3. blok (`rwx`): uprawnienia użytkowników w grupie właściciela,
- 4. blok (`rwx`): uprawnienia wszystkich pozostałych użytkowników.

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~/zadanie1$ ls -l main.c
-rw-r--r-- 1 student student 0 wrz 1 23:20 main.c
student@wftlab-180:~/zadanie1$
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~/zadanie1$ ls -l main.c
-rw-r--r-- 1 student student 0 wrz 1 23:20 main.c
student@wftlab-180:~/zadanie1$ chmod
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~/zadanie1$ ls -l main.c
-rw-r--r-- 1 student student 0 wrz 1 23:20 main.c
student@wftlab-180:~/zadanie1$ chmod -r
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~/zadanie1$ ls -l main.c
-rw-r--r-- 1 student student 0 wrz 1 23:20 main.c
student@wftlab-180:~/zadanie1$ chmod -r main.c
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~/zadanie1$ ls -l main.c
-rw-r--r-- 1 student student 0 wrz 1 23:20 main.c
student@wftlab-180:~/zadanie1$ chmod -r main.c
student@wftlab-180:~/zadanie1$
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~/zadanie1$ ls -l main.c
-rw-r--r-- 1 student student 0 wrz 1 23:20 main.c
student@wftlab-180:~/zadanie1$ chmod -r main.c
student@wftlab-180:~/zadanie1$
```

Brak komunikatu oznacza: OK – zrobione!

```
student@wftlab-180:~/zadanie1$
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~/zadanie1$ ls -l main.c
-rw-r--r-- 1 student student 0 wrz 1 23:20 main.c
student@wftlab-180:~/zadanie1$ chmod -r main.c
student@wftlab-180:~/zadanie1$
```

Brak komunikatu oznacza: OK – zrobione!

```
student@wftlab-180:~/zadanie1$ ls -l main.c
```

System operacyjny GNU/Linux: **CLI**

```
student@wftlab-180:~/zadanie1$ ls -l main.c
-rw-r--r-- 1 student student 0 wrz 1 23:20 main.c
student@wftlab-180:~/zadanie1$ chmod -r main.c
student@wftlab-180:~/zadanie1$
```

Brak komunikatu oznacza: OK – zrobione!

```
student@wftlab-180:~/zadanie1$ ls -l main.c
--w------- 1 student student 0 wrz 1 23:20 main.c
student@wftlab-180:~/zadanie1$
```

```
student@wftlab-180:~/zadanie1$ ls -l main.c
-rw-r--r-- 1 student student 0 wrz 1 23:20 main.c
student@wftlab-180:~/zadanie1$ chmod -r main.c
student@wftlab-180:~/zadanie1$
```

Brak komunikatu oznacza: OK – zrobione!

```
student@wftlab-180:~/zadanie1$ ls -l main.c
--w------- 1 student student 0 wrz 1 23:20 main.c
student@wftlab-180:~/zadanie1$
```

Uruchamianie skryptów:

tworząc nowy skrypt będziemy najczęściej nadawać mu prawo wykonywalności, choć nie jest to konieczne (o tym później).

Edycja tekstu: `vim` *vi improved*

```
\section{Programowanie w środowisku GNU/Linux}
\subsection{Wprowadzenie}

\frame{
\frametitle{Języki kompilowane i interpretowane}

\begin{block}{Jeden z możliwych podziałów języków programowania pozwala wyodrębnić dwie (niejednoznaczne) kategorie:}
\begin{itemize}
\item języki \textbf{kompilowane} -- kod źródłowy jest tłumaczony na kod maszynowy za pomocą kompilatora (ANSI C, C++, FORTRAN i in.),
\item języki \textbf{interpretowane} -- kod źródłowy jest wykonywany bezpośrednio przez interpreter (bash, Python, Perl, PHP i in.).
\end{itemize}
\end{block}

\pause

\begin{exampleblock}{Narzędzia programistyczne:}
\begin{itemize}
\item edytor tekstu, kompilator (w tym linker)/interpreter, debugger,
\item opcjonalnie: środowisko programistyczne zawierające funkcjonalność wszystkich powyższych narzędzi (i więcej).
\end{itemize}
\end{exampleblock}

\small
Przykładowe narzędzia:
\begin{itemize}
\item vim, emacs (edytory),
\item gcc, gfortran (kompilatory); gdb (debugger).
\end{itemize}
}
```

- Edytor tekstu wydany na wiele plaform systemowych; bogactwo możliwości edycyjnych; wygodne *środowisko* programistyczne w połączeniu z kompilatorem gcc i narzędziami powłoki;
- wiele trybów edycji (podstawowa obsługa na zajęciach w pracowni).

Edycja tekstu: vim *vi improved*

```
$ vim nazwa_pliku
```

- `:w nazwa` – zapis do pliku,
- `:wq` – zapis i wyjście,
- `:q!` – wyjście (ignoruj zmiany),
- `i` – tryb edycji,
- `[ESC]` – opuść tryb edycji,
- `u` – cofnij,
- `<Ctrl-R>` – powtórz,
- `/wzorzec` – szukaj wzorca (`n` – nast. / `N` – poprzedni),
- `:!polecenie` – polec. powłoki,
- `dd` – usuń bieżący wiersz,
- `D` – usuń stąd do końca wiersza,
- `:%s/raz/dwa/g` – znajdź i zastąp (*raz*→*dwa*),
- `:s/raz/dwa/g` – znajdź i zastąp w bieżącym wierszu,
- `:s/raz/dwa/gc` – znajdź i zastąp z potwierdzeniem,
- `v` – zaznacz (przesuwając kursor: `[←]`, `[↑]`, `[↓]`, `[→]`),
- `y` – skopiuj zaznaczenie,
- `d` – wytnij zaznaczenie,
- `p` – wklej skopiowane.

Uruchamianie skryptów powłoki

```
$ ./hello.sh
```

Ale:

musi być wykonywalny
(`chmod +x hello.sh`).

```
$ bash ./hello.sh
```

Wówczas:

nie musi być wykonywalny
(wystarczy `-r--------`).

Uruchamianie skryptów powłoki

```
$ ./hello.sh
```

Ale:

musi być wykonywalny (`chmod +x hello.sh`).

```
$ bash ./hello.sh
```

Wówczas:

nie musi być wykonywalny (wystarczy `-r--------`).

Standardowe wyjście (*stdout*, 1)

```
$ ./hello.sh
Hello World!
```

← standardowe WYjście

Uruchamianie skryptów powłoki

```
$ ./hello.sh
```

Ale:

musi być wykonywalny
(`chmod +x hello.sh`).

```
$ bash ./hello.sh
```

Wówczas:

nie musi być wykonywalny
(wystarczy `-r--------`).

Standardowe wyjście (*stdout*, 1)

```
$ ./hello.sh
Hello World!
```
← standardowe WYjście

Przekierowanie *stdout* do pliku:

```
$ ./hello.sh > plik
```

Uruchamianie skryptów powłoki

```
$ ./hello.sh
```

```
$ bash ./hello.sh
```

Ale:

musi być wykonywalny (`chmod +x hello.sh`).

Wówczas:

nie musi być wykonywalny (wystarczy `-r--------`).

Standardowe wyjście (*stdout*, 1)

```
$ ./hello.sh
Hello World!
```
← standardowe WYjście

Przekierowanie *stdout* do pliku:

```
$ ./hello.sh > plik
```

Alternatywnie:

```
$ ./hello.sh 1>plik
```

Uruchamianie skryptów powłoki

Analogicznie (*stderr* = 2):

```
$ ./hello.sh 2>plik_err
```

Uruchamianie skryptów powłoki

Analogicznie (*stderr* = 2):

```
$ ./hello.sh 2>plik_err
```

stderr → *stdout*:

```
$ ./hello.sh 2>&1
```

Uruchamianie skryptów powłoki

Analogicznie (*stderr* = 2):

```
$ ./hello.sh 2>plik_err
```

stderr → *stdout*:

```
$ ./hello.sh 2>&1
```

stdout i *stderr* do **jednego** pliku:

```
$ ./hello.sh &>plik
```

Uruchamianie skryptów powłoki

Analogicznie (*stderr* = 2):

```
$ ./hello.sh 2>plik_err
```

stderr → *stdout*:

```
$ ./hello.sh 2>&1
```

stdout i *stderr* do **jednego** pliku:

```
$ ./hello.sh &>plik
```

Można też:

```
$ ./hello.sh &>/dev/null
```

Uruchamianie skryptów powłoki

Analogicznie (*stderr* = 2):

```
$ ./hello.sh 2>plik_err
```

stderr → *stdout*:

```
$ ./hello.sh 2>&1
```

stdout i *stderr* do **jednego** pliku:

```
$ ./hello.sh &>plik
```

Można też:

```
$ ./hello.sh &>/dev/null
```

A także:

```
$ ./hello.sh 1>plik 2>bledy
```

System modułów aplikacji

Environment Modules (`https://modules.readthedocs.io/en/v4.1.4`)

The Modules package is a tool that simplify shell initialization and lets users easily modify their environment during the session with modulefiles.

Each modulefile contains the information needed to configure the shell for an application (...) Typically modulefiles instruct the module command to alter or set shell environment variables such as PATH, MANPATH, etc. modulefiles may be shared by many users on a system and users may have their own collection to supplement or replace the shared modulefiles.

Modules can be loaded and unloaded dynamically (...) Modules are useful in managing different versions of applications.

System modułów aplikacji

Uproszczony plik modułu:

```
#%Module1.0
module-whatis ‘‘This module loads Abinit 8.10.3’’
module load common/compilers/intel
prereq common/compilers/intel
set VERSION 8.10.3
set PREFIX /apps/abinit
set APP_DIR $PREFIX/$VERSION/INTEL/bin
prepend-path PATH $APP_DIR
```

System modułów aplikacji

- `module avail`
- `module load ...`
- `module unload ...`
- `module display ...`
- `module` – dostępne opcje

System modułów aplikacji

- `module avail`
- `module load ...`
- `module unload ...`
- `module display ...`
- `module` – dostępne opcje

Lokalne moduły użytkownika:

- `module use /path/to/personal/modulefiles`

System modułów aplikacji

- `module avail`
- `module load ...`
- `module unload ...`
- `module display ...`
- `module` – dostępne opcje

Lokalne moduły użytkownika:

- `module use /path/to/personal/modulefiles`

Można też:

- `module load use.own`

System modułów aplikacji

Dostępne aplikacje:

https://kdm.icm.edu.pl/Aplikacje/aplikacje.pl

Status CI:

https://kdm.icm.edu.pl/Aplikacje/ci_status.pl

System modułów aplikacji

Dostępne aplikacje:

https://kdm.icm.edu.pl/Aplikacje/aplikacje.pl

Status CI:

https://kdm.icm.edu.pl/Aplikacje/ci_status.pl

Czy możemy dodać aplikację X?

Kompilacja własnego oprogramowania

Topola: Dostępne kompilatory:

Programy sekwencyjne:

- Intel: `module load common/compilers/intel`
- GNU: `module load common/compilers/gnu`

Programy równoległe (MPI) – *wrappery* `mpicc/mpif90`

- Intel + MPICH (zalecane): `module load common/mpi/mpich/3.3.1`
- GNU + OpenMPI: `module load common/mpi/openmpi/3.0.0`

Kompilacja własnego oprogramowania

Okeanos: Dostępne kompilatory:

Programy sekwencyjne:

- Cray (domyślnie) (`cc`)
- Intel: `module load intel` (`icc/ifort`)
- GNU: `module load gnu` (`gcc/gfortran`)

Programy równoległe (MPI) – *wrappery* `cc/CC/ftn`

- Cray (domyślnie)
- Intel: `module swap PrgEnv-cray PrgEnv-intel`
- GNU: `module swap PrgEnv-cray PrgEnv-gnu`

Slurm: System kolejkowy

Queue system for supercomputers and clusters

- Alokuje zasoby (węzły, CPU, pamięć operacyjna) dla zadań obliczeniowych,
- uruchamia zadania równoległe na zaalokowanych zasobach,
- dba o równomierne obciążenie maszyn i równy przydział zasobów.

Slurm: System kolejkowy

Queue system for supercomputers and clusters

- Alokuje zasoby (węzły, CPU, pamięć operacyjna) dla zadań obliczeniowych,
- uruchamia zadania równoległe na zaalokowanych zasobach,
- dba o równomierne obciążenie maszyn i równy przydział zasobów.

Pojęcia:

- węzeł (ang. *node*) – pojedyncza jednostka obliczeniowa wyposażona w CPU i pamięć),
- partycja (and. *partition* – zbiór węzłów o określonej charakterystyce,
- zadanie (ang. *job*) – pojedyncze zadanie obliczeniowe.

Slurm: System kolejkowy

Queue system for supercomputers and clusters

- Alokuje zasoby (węzły, CPU, pamięć operacyjna) dla zadań obliczeniowych,
- uruchamia zadania równoległe na zaalokowanych zasobach,
- dba o równomierne obciążenie maszyn i równy przydział zasobów.

Pojęcia:

- węzeł (ang. *node*) – pojedyncza jednostka obliczeniowa wyposażona w CPU i pamięć),
- partycja (and. *partition* – zbiór węzłów o określonej charakterystyce,
- zadanie (ang. *job*) – pojedyncze zadanie obliczeniowe.

Przykłady nazw partycji: *topola*, *okeanos* (domyślna), *gpu*, *ve*.

Slurm: System kolejkowy

Wybrane instrukcje:

- `sbatch` – umieszcza zadanie (skrypt zadania) w kolejce (alokuje zasoby),
- `srun` – uruchamia zadanie równoległe w ramach alokacji,
- `scancel` – usuwa zadanie z kolejki,
- `squeue` – wyświetla stan kolejki,
- `sinfo` – wyświetla informacje o węzłach i partycjach,
- `scontrol` – wyświetla szczegóły (lub modyfikuje) zadanie.

Slurm – System kolejkowy

Sesja interaktywna:

```
srun -A nr-alokacji -p topola --nodes=1 --ntasks-per-node=1 --pty bash -l
```

Slurm – System kolejkowy

Sesja interaktywna:

```
srun -A nr-alokacji -p topola --nodes=1 --ntasks-per-node=1 --pty bash -l
```

Tryb wsadowy – przykładowy skrypt:

```
#!/bin/bash -l
#SBATCH -J nazwa
#SBATCH --nodes 1
#SBATCH --ntasks-per-node 1
#SBATCH --mem 1000
#SBATCH --time=1:00:00
#SBATCH -A <Grant_ID>
#SBATCH -p topola
./program
```

Slurm – System kolejkowy

Sesja interaktywna:

```
srun -A nr-alokacji -p topola --nodes=1 --ntasks-per-node=1 --pty bash -l
```

Tryb wsadowy – przykładowy skrypt:

```
#!/bin/bash -l
#SBATCH -J nazwa
#SBATCH --nodes 1
#SBATCH --ntasks-per-node 1
#SBATCH --mem 1000
#SBATCH --time=1:00:00
#SBATCH -A <Grant_ID>
#SBATCH -p topola
./program
```

Zlecenie zadania:

```
sbatch job.sl
```

Anulowanie zadania:

```
scancel 1234567
```

Szczegóły zadania:

```
scontrol show job 1234567
```

Slurm – System kolejkowy

Listing zadań w kolejce (squeue):

```
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
5328766 topola hfor_LaB mariana PD 0:00 10 (Resources)
5328767 topola hfor_LaB mariana PD 0:00 10 (Priority)
5328768 topola hfor_LaB mariana PD 0:00 10 (Priority)
5328769 topola hfor_LaB mariana PD 0:00 10 (Priority)
5329499 topola snpk8_1 purpurea R 9:46:09 1 t7-15
5329498 topola snpk7_5 purpurea R 9:46:32 1 t7-15
5329497 topola snpk7_4 purpurea R 9:46:39 1 t7-15
5329496 topola snpk7_3 purpurea R 9:46:45 1 t7-15
5328765 topola hfor_LaB mariana R 30:01 10 t9-[7-8,12],t10-[2-3]
```

Sesja praktyczna / demonstracje

- Logowanie i uruchamianie zadań obliczeniowych,
- system modułów, system kolejkowy,
- kopiowanie danych między systemami,
- Python – środowisko wirtualne.