

# Podstawy informatyki

WYKŁAD nr 03

Fizyka Techniczna, WFT PP

Michał Hermanowicz

Zakład Fizyki Obliczeniowej, Instytut Fizyki, Politechnika Poznańska

Rok akademicki 2018/2019



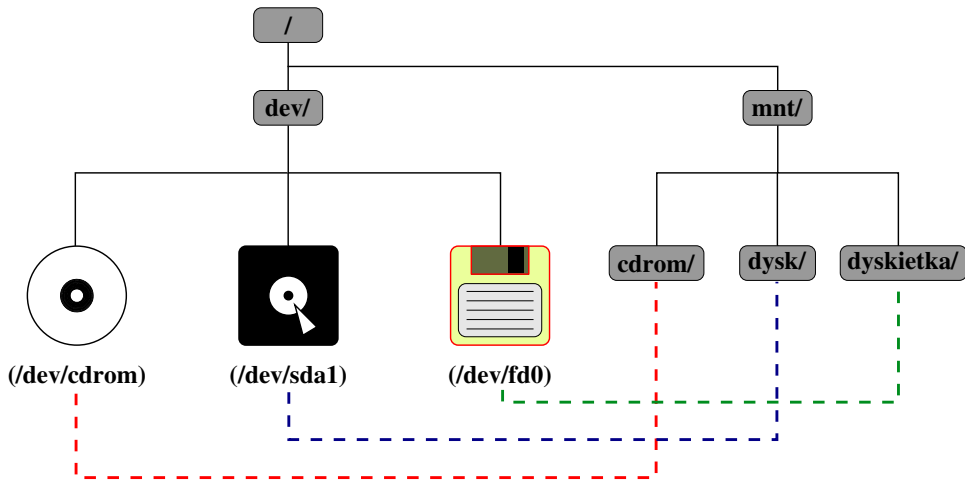
- 1 Przypomnienie z poprzedniego wykładu
- 2 Zarządzanie danymi w systemie GNU/Linux
- 3 Standardy przechowywania danych
- 4 Operacje na danych
- 5 Graficzna reprezentacja danych – gnuplot
  - Wykresy 2D i regresja liniowa
- 6 Skrypty i procesy
- 7 Podsumowanie: pytania i dyskusja

# Plan ramowy przedmiotu

Nr wykładu	Poruszane zagadnienia
I	Organizacja; forma i warunki zaliczenia; wprowadzenie
II	Powłoka <b>bash</b> i elementy programowania
III	Przetwarzanie danych #1
IV	Przetwarzanie danych #2
V	Reprezentacja danych (wykresy 2D i 3D) – <i>gnuplot</i>
VI	System składu tekstu L <sup>A</sup> T <sub>E</sub> X
VII	Pół-otwarty test zaliczeniowy

Każdemu z wykładów odpowiadają ćwiczenia realizowane na zajęciach w pracowni komputerowej.

# Organizacja i nośniki danych



## Aby użyć systemu plików w GNU/Linuxie:

- podłączamy nośnik danych do komputera (*pendrive*, nowy dysk twardy, napęd CD-ROM),

# Pliki urządzeń i *montowanie* systemu plików

## Aby użyć systemu plików w GNU/Linuxie:

- podłączamy nośnik danych do komputera (*pendrive*, nowy dysk twardy, napęd CD-ROM),
- nowe urządzenie jest plikiem (`/dev/nazwa`), którego jednak nie można użyć bezpośrednio – trzeba je uprzednio *zamontować*.

# Pliki urządzeń i *montowanie* systemu plików

## Aby użyć systemu plików w GNU/Linuxie:

- podłączamy nośnik danych do komputera (*pendrive*, nowy dysk twardy, napęd CD-ROM),
- nowe urządzenie jest plikiem (`/dev/nazwa`), którego jednak nie można użyć bezpośrednio – trzeba je uprzednio *zamontować*.

## *Montowanie* nośnika danych:

# Pliki urządzeń i *montowanie* systemu plików

## Aby użyć systemu plików w GNU/Linuxie:

- podłączamy nośnik danych do komputera (*pendrive*, nowy dysk twardy, napęd CD-ROM),
- nowe urządzenie jest plikiem (`/dev/nazwa`), którego jednak nie można użyć bezpośrednio – trzeba je uprzednio *zamontować*.

## *Montowanie* nośnika danych:

- czynność, w wyniku której nośnik staje się zdalny do odczytu/zapisu,



# Pliki urządzeń i *montowanie* systemu plików

## Aby użyć systemu plików w GNU/Linuxie:

- podłączamy nośnik danych do komputera (*pendrive*, nowy dysk twardy, napęd CD-ROM),
- nowe urządzenie jest plikiem (`/dev/nazwa`), którego jednak nie można użyć bezpośrednio – trzeba je uprzednio *zamontować*.

## *Montowanie* nośnika danych:

- czynność, w wyniku której nośnik staje się zdatny do odczytu/zapisu,
- oznacza umieszczanie go w istniejącej hierarchii plików w formie katalogu, z którego można już bezpośrednio odczytywać dane i do którego można dane zapisywać. Np. urządzenie `/dev/sdb1` może zostać *zamontowane* jako katalog `/mnt/pendrive` (lub dowolny inny) – jest to tzw. *punkt montowania*, który możemy dowolnie zadać.

# Polecenie mount

Składnia (man mount):

```
mount -t [typ] [urządzenie] [punkt montowania]
```

gdzie:

[typ] – system plików (ext4, ntfs, vfat i in.),

[urządzenie] – plik urządzenia (np. /dev/cdrom, /dev/sdb1),

[punkt montowania] – katalog w istniejącej hierarchii plików.

# Polecenie mount

Składnia (man mount):

```
mount -t [typ] [urządzenie] [punkt montowania]
```

gdzie:

[typ] – system plików (ext4, ntfs, vfat i in.),

[urządzenie] – plik urządzenia (np. /dev/cdrom, /dev/sdb1),

[punkt montowania] – katalog w istniejącej hierarchii plików.

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

# Polecenie mount

Składnia (man mount):

```
mount -t [typ] [urządzenie] [punkt montowania]
```

gdzie:

[typ] – system plików (ext4, ntfs, vfat i in.),

[urządzenie] – plik urządzenia (np. /dev/cdrom, /dev/sdb1),

[punkt montowania] – katalog w istniejącej hierarchii plików.

```
student@wftlab-180:~$ mount -t vfat /dev/sdb1 /mnt/pendrive
```

# Polecenie mount

Składnia (man mount):

```
mount -t [typ] [urządzenie] [punkt montowania]
```

gdzie:

[typ] – system plików (ext4, ntfs, vfat i in.),

[urządzenie] – plik urządzenia (np. /dev/cdrom, /dev/sdb1),

[punkt montowania] – katalog w istniejącej hierarchii plików.

```
student@wftlab-180:~$ mount -t vfat /dev/sdb1 /mnt/pendrive
```

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

# Polecenie mount

Składnia (man mount):

```
mount -t [typ] [urządzenie] [punkt montowania]
```

gdzie:

[typ] – system plików (ext4, ntfs, vfat i in.),

[urządzenie] – plik urządzenia (np. /dev/cdrom, /dev/sdb1),

[punkt montowania] – katalog w istniejącej hierarchii plików.

```
student@wftlab-180:~$ mount -t vfat /dev/sdb1 /mnt/pendrive
```

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

# Polecenie mount

Składnia (man mount):

```
mount -t [typ] [urządzenie] [punkt montowania]
```

gdzie:

[typ] – system plików (ext4, ntfs, vfat i in.),

[urządzenie] – plik urządzenia (np. /dev/cdrom, /dev/sdb1),

[punkt montowania] – katalog w istniejącej hierarchii plików.

```
student@wftlab-180:~$ mount -t vfat /dev/sdb1 /mnt/pendrive
```

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

System plików	rozm.	użyte	dost.	%uż.	zamont. na
/dev/sdb1	8,0G	0	8,0G	0%	/mnt/pendrive

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

# Polecenie mount

Składnia (man mount):

```
mount -t [typ] [urządzenie] [punkt montowania]
```

gdzie:

[typ] – system plików (ext4, ntfs, vfat i in.),

[urządzenie] – plik urządzenia (np. /dev/cdrom, /dev/sdb1),

[punkt montowania] – katalog w istniejącej hierarchii plików.

```
student@wftlab-180:~$ mount -t vfat /dev/sdb1 /mnt/pendrive
```

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

System plików	rozm.	użyte	dost.	%uż.	zamont. na
/dev/sdb1	8,0G	0	8,0G	0%	/mnt/pendrive

```
student@wftlab-180:~$ cp dane /mnt/pendrive
```



# Polecenie mount

Składnia (man mount):

```
mount -t [typ] [urządzenie] [punkt montowania]
```

gdzie:

[typ] – system plików (ext4, ntfs, vfat i in.),

[urządzenie] – plik urządzenia (np. /dev/cdrom, /dev/sdb1),

[punkt montowania] – katalog w istniejącej hierarchii plików.

```
student@wftlab-180:~$ mount -t vfat /dev/sdb1 /mnt/pendrive
```

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

System plików	rozm.	użyte	dost.	%uż.	zamont. na
/dev/sdb1	8,0G	0	8,0G	0%	/mnt/pendrive

```
student@wftlab-180:~$ cp dane /mnt/pendrive
```

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

# Polecenie mount

Składnia (man mount):

```
mount -t [typ] [urządzenie] [punkt montowania]
```

gdzie:

[typ] – system plików (ext4, ntfs, vfat i in.),

[urządzenie] – plik urządzenia (np. /dev/cdrom, /dev/sdb1),

[punkt montowania] – katalog w istniejącej hierarchii plików.

```
student@wftlab-180:~$ mount -t vfat /dev/sdb1 /mnt/pendrive
```

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

System plików	rozm.	użyte	dost.	%uż.	zamont. na
/dev/sdb1	8,0G	0	8,0G	0%	/mnt/pendrive

```
student@wftlab-180:~$ cp dane /mnt/pendrive
```

```
student@wftlab-180:~$ umount /mnt/pendrive
```

# 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/
```

## SSH (*Secure Shell*):

protokół sieciowy umożliwiający bezpieczną (szyfrowane połączenie) pracę na zdalnych komputerach.



## SSH (*Secure Shell*):

protokół sieciowy umożliwiający bezpieczną (szyfrowane połączenie) pracę na zdalnych komputerach.

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

## SSH (*Secure Shell*):

protokół sieciowy umożliwiający bezpieczną (szyfrowane połączenie) pracę na zdalnych komputerach.

```
student@wftlab-180:~$ ssh uzytkownik@serwer.pl
```

## SSH (*Secure Shell*):

protokół sieciowy umożliwiający bezpieczną (szyfrowane połączenie) pracę na zdalnych komputerach.

```
student@wftlab-180:~$ ssh uzytkownik@serwer.pl  
uzytkownik@serwer.pl's password:
```

## SSH (*Secure Shell*):

protokół sieciowy umożliwiający bezpieczną (szyfrowane połączenie) pracę na zdalnych komputerach.

```
student@wftlab-180:~$ ssh uzytkownik@serwer.pl
uzytkownik@serwer.pl's password:
uzytkownik@serwer.pl:~$
```

## SSH (*Secure Shell*):

protokół sieciowy umożliwiający bezpieczną (szyfrowane połączenie) pracę na zdalnych komputerach.

```
student@wftlab-180:~$ ssh uzytkownik@serwer.pl
uzytkownik@serwer.pl's password:
uzytkownik@serwer.pl:~$
uzytkownik@serwer.pl:~$
```

## SSH (*Secure Shell*):

protokół sieciowy umożliwiający bezpieczną (szyfrowane połączenie) pracę na zdalnych komputerach.

```
student@wftlab-180:~$ ssh uzytkownik@serwer.pl
uzytkownik@serwer.pl's password:
uzytkownik@serwer.pl:~$
uzytkownik@serwer.pl:~$ exit
```

## SSH (*Secure Shell*):

protokół sieciowy umożliwiający bezpieczną (szyfrowane połączenie) pracę na zdalnych komputerach.

```
student@wftlab-180:~$ ssh uzytkownik@serwer.pl
uzytkownik@serwer.pl's password:
uzytkownik@serwer.pl:~$
uzytkownik@serwer.pl:~$ exit
student@wftlab-180:~$
```

## SSH (*Secure Shell*):

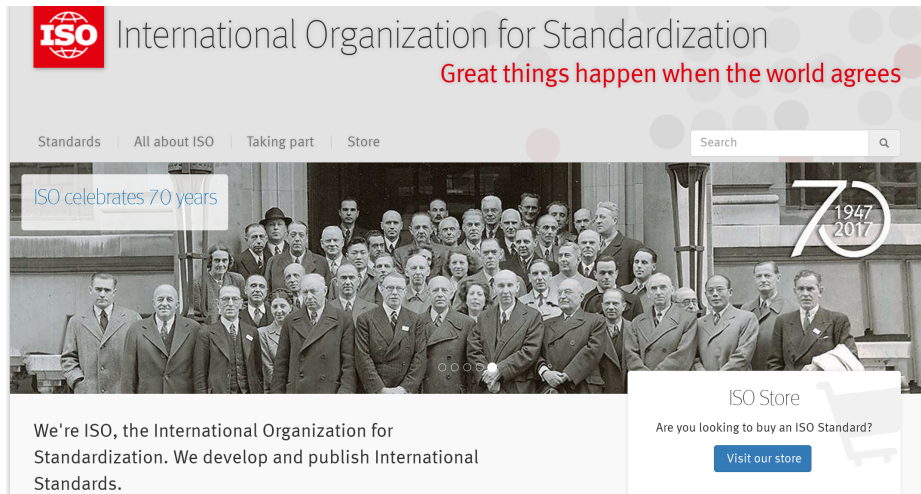
protokół sieciowy umożliwiający bezpieczną (szyfrowane połączenie) pracę na zdalnych komputerach.

```
student@wftlab-180:~$ ssh uzytkownik@serwer.pl
uzytkownik@serwer.pl's password:
uzytkownik@serwer.pl:~$
uzytkownik@serwer.pl:~$ exit
student@wftlab-180:~$
```

## SCP (*Secure Copy Protocol*):

```
scp ./plik herman@serwer.pl:/home/herman/
```





The banner features the ISO logo (a red square with a white globe icon and the letters 'ISO') in the top left. To its right is the text 'International Organization for Standardization' in a large, dark font, followed by the tagline 'Great things happen when the world agrees' in a smaller, red font. Below this is a navigation bar with links: 'Standards', 'All about ISO', 'Taking part', and 'Store'. A search bar with the placeholder text 'Search' and a magnifying glass icon is on the right. The main visual is a large black and white photograph of a group of men in suits, likely the founding members of ISO. Overlaid on the left of the photo is a white box with the text 'ISO celebrates 70 years'. On the right of the photo is a large white '70' with '1947' and '2017' inside the loops. At the bottom left, a white box contains the text: 'We're ISO, the International Organization for Standardization. We develop and publish International Standards.' At the bottom right, a white box contains the text 'ISO Store' and 'Are you looking to buy an ISO Standard?' with a blue button labeled 'Visit our store' and a faint shopping cart icon.

ISO International Organization for Standardization

Great things happen when the world agrees

Standards | All about ISO | Taking part | Store

Search

ISO celebrates 70 years

1947 2017

We're ISO, the International Organization for Standardization. We develop and publish International Standards.

ISO Store

Are you looking to buy an ISO Standard?

Visit our store

# Standard a 'de-facto' standard

## Standard:

- ustalony i powszechnie zaakceptowany zbiór zasad (norm) określających sposób działania/wytwarzania; stosowany w celu zapewnienia zgodności.

## 'De-facto' standard:

- zwyczajowo przyjęty zbiór zasad (norm) określających sposób działania/wytwarzania; stosowany na podstawie przyjętego zwyczaju.

# Standard a 'de-facto' standard

## Standard:

- ustalony i powszechnie zaakceptowany zbiór zasad (norm) określających sposób działania/wytwarzania; stosowany w celu zapewnienia zgodności.

## 'De-facto' standard:

- zwyczajowo przyjęty zbiór zasad (norm) określających sposób działania/wytwarzania; stosowany na podstawie przyjętego zwyczaju.

Każdy standard, co do zasady, musi być otwarty, tzn. mieć jawną specyfikację.

# Przetwarzanie danych



## INPUT (wejście):

- plik (dowolnego typu),
- strumień danych (również pochodzący z *potoku*).

## PRZETWARZANIE:

- program/skrypt wykonujący operacje na danych wejściowych.

## OUTPUT (wyjście):

- przetworzone dane (zapisane do pliku lub na standardowe wyjście).

Plik tekstowy liczby.txt:

1234	7564	3761	4176	8786
2456	5465	8361	1112	7711
8462	2324	5545	3332	3471
7568	3456	1142	6161	4221
5547	5091	8181	4444	8123

# Przetwarzanie danych tekstowych

Plik tekstowy liczby.txt:

1234	7564	3761	4176	8786
2456	5465	8361	1112	7711
8462	2324	5545	3332	3471
7568	3456	1142	6161	4221
5547	5091	8181	4444	8123

student@wftlab-180:~\$

# Przetwarzanie danych tekstowych

Plik tekstowy liczby.txt:

```
1234 7564 3761 4176 8786
2456 5465 8361 1112 7711
8462 2324 5545 3332 3471
7568 3456 1142 6161 4221
5547 5091 8181 4444 8123
```

```
student@wftlab-180:~$ grep 9 liczby.txt
```

# Przetwarzanie danych tekstowych

Plik tekstowy liczby.txt:

```
1234 7564 3761 4176 8786
2456 5465 8361 1112 7711
8462 2324 5545 3332 3471
7568 3456 1142 6161 4221
5547 5091 8181 4444 8123
```

```
student@wftlab-180:~$ grep 9 liczby.txt
```

```
5547 5091 8181 4444 8123
```

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



# Przetwarzanie danych tekstowych

Plik tekstowy liczby.txt:

```
1234 7564 3761 4176 8786
2456 5465 8361 1112 7711
8462 2324 5545 3332 3471
7568 3456 1142 6161 4221
5547 5091 8181 4444 8123
```

```
student@wftlab-180:~$ grep 9 liczby.txt
```

```
5547 5091 8181 4444 8123
```

```
student@wftlab-180:~$ cat liczby.txt | grep 9
```

# Przetwarzanie danych tekstowych

Plik tekstowy liczby.txt:

```
1234 7564 3761 4176 8786
2456 5465 8361 1112 7711
8462 2324 5545 3332 3471
7568 3456 1142 6161 4221
5547 5091 8181 4444 8123
```

```
student@wftlab-180:~$ grep 9 liczby.txt
```

```
5547 5091 8181 4444 8123
```

```
student@wftlab-180:~$ cat liczby.txt | grep 9
```

```
5547 5091 8181 4444 8123
```

```
student@wftlab-180:~$ cat liczby.txt | grep 9 | wc -l
```

# Przetwarzanie danych tekstowych

Plik tekstowy liczby.txt:

```
1234  7564  3761  4176  8786
2456  5465  8361  1112  7711
8462  2324  5545  3332  3471
7568  3456  1142  6161  4221
5547  5091  8181  4444  8123
```

```
student@wftlab-180:~$ grep 9 liczby.txt
```

```
5547  5091  8181  4444  8123
```

```
student@wftlab-180:~$ cat liczby.txt | grep 9
```

```
5547  5091  8181  4444  8123
```

```
student@wftlab-180:~$ cat liczby.txt | grep 9 | wc -l
```

```
1
```

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

# Przetwarzanie danych tekstowych (man cut)

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

## Przetwarzanie danych tekstowych (man cut)

```
student@wftlab-180:~$ cut -d ' ' -f 1 liczby.txt
```

## Przetwarzanie danych tekstowych (man cut)

```
student@wftlab-180:~$ cut -d ' ' -f 1 liczby.txt
```

```
1234
```

```
2456
```

```
8462
```

```
7568
```

```
5547
```

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

## Przetwarzanie danych tekstowych (man cut)

```
student@wftlab-180:~$ cut -d ' ' -f 1 liczby.txt
```

1234

2456

8462

7568

5547

```
student@wftlab-180:~$ cut -d ' ' -f1,2 liczby.txt
```

## Przetwarzanie danych tekstowych (man cut)

```
student@wftlab-180:~$ cut -d ' ' -f 1 liczby.txt
```

1234

2456

8462

7568

5547

```
student@wftlab-180:~$ cut -d ' ' -f1,2 liczby.txt
```

1234 7564

2456 5465

8462 2324

7568 3456

5547 5091

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



# Przetwarzanie danych tekstowych (man awk; man sed)

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

## Przetwarzanie danych tekstowych (man awk; man sed)

```
student@wftlab-180:~$ awk '{print $1}' liczby.txt
```

## Przetwarzanie danych tekstowych (man awk; man sed)

```
student@wftlab-180:~$ awk '{print $1}' liczby.txt
```

```
1234
```

```
2456
```

```
8462
```

```
7568
```

```
5547
```

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

## Przetwarzanie danych tekstowych (man awk; man sed)

```
student@wftlab-180:~$ awk '{print $1}' liczby.txt
```

1234

2456

8462

7568

5547

```
student@wftlab-180:~$ cat liczby.txt | sed 's/5/1/g'
```

## Przetwarzanie danych tekstowych (man awk; man sed)

```
student@wftlab-180:~$ awk '{print $1}' liczby.txt
```

1234

2456

8462

7568

5547

```
student@wftlab-180:~$ cat liczby.txt | sed 's/5/1/g'
```

1234 7164 3761 4176 8786

2416 1461 8361 1112 7711

8462 2324 1141 3332 3471

7168 3416 1142 6161 4221

1147 1091 8181 4444 8123

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

# Przetwarzanie danych tekstowych (man tr)

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

## Przetwarzanie danych tekstowych (man tr)

```
student@wftlab-180:~$ cat liczby.txt | tr -s '\n' ' '
```

## Przetwarzanie danych tekstowych (man tr)

```
student@wftlab-180:~$ cat liczby.txt | tr -s '\n' ' '
1234 7564 3761 4176 8786 2456 5465 8361 1112 7711 8462 2324 5545 3332 3471
7568 3456 1142 6161 4221 5547 5091 8181 4444 8123
student@wftlab-180:~$
```



## Przetwarzanie danych tekstowych (man tr)

```
student@wftlab-180:~$ cat liczby.txt | tr -s '\n' ' '
1234 7564 3761 4176 8786 2456 5465 8361 1112 7711 8462 2324 5545 3332 3471
7568 3456 1142 6161 4221 5547 5091 8181 4444 8123
student@wftlab-180:~$ tr -s '\n' ' ' < liczby.txt
```

## Przetwarzanie danych tekstowych (man tr)

```
student@wftlab-180:~$ cat liczby.txt | tr -s '\n' ' '
1234 7564 3761 4176 8786 2456 5465 8361 1112 7711 8462 2324 5545 3332 3471
7568 3456 1142 6161 4221 5547 5091 8181 4444 8123
student@wftlab-180:~$ tr -s '\n' ' ' < liczby.txt
1234 7564 3761 4176 8786 2456 5465 8361 1112 7711 8462 2324 5545 3332 3471
7568 3456 1142 6161 4221 5547 5091 8181 4444 8123
student@wftlab-180:~$
```

### Alternatywnie:

```
student@wftlab-180:~$ awk 'printf("%s ", $0)' liczby.txt
```

## Przetwarzanie danych tekstowych (man tr)

```
student@wftlab-180:~$ cat liczby.txt | tr -s '\n' ' '
1234 7564 3761 4176 8786 2456 5465 8361 1112 7711 8462 2324 5545 3332 3471
7568 3456 1142 6161 4221 5547 5091 8181 4444 8123
student@wftlab-180:~$ tr -s '\n' ' ' < liczby.txt
1234 7564 3761 4176 8786 2456 5465 8361 1112 7711 8462 2324 5545 3332 3471
7568 3456 1142 6161 4221 5547 5091 8181 4444 8123
student@wftlab-180:~$
```

### Alternatywnie:

```
student@wftlab-180:~$ awk 'printf("%s ", $0)' liczby.txt
student@wftlab-180:~$ for row in `cat liczby.txt` ; do echo -en $row ; done
```

# Przetwarzanie danych tekstowych (man tr)

```
student@wftlab-180:~$ cat liczby.txt | tr -s '\n' ' '
1234 7564 3761 4176 8786 2456 5465 8361 1112 7711 8462 2324 5545 3332 3471
7568 3456 1142 6161 4221 5547 5091 8181 4444 8123
student@wftlab-180:~$ tr -s '\n' ' ' < liczby.txt
1234 7564 3761 4176 8786 2456 5465 8361 1112 7711 8462 2324 5545 3332 3471
7568 3456 1142 6161 4221 5547 5091 8181 4444 8123
student@wftlab-180:~$
```

## Alternatywnie:

```
student@wftlab-180:~$ awk 'printf("%s ", $0)' liczby.txt
student@wftlab-180:~$ for row in `cat liczby.txt` ; do echo -en $row ; done
```

**To jest ważne!**

find – wyszukiwanie plików/katalogów (man find)

```
student@wftlab-180:~$ find ./ -name *.txt
```

find – wyszukiwanie plików/katalogów (man find)

```
student@wftlab-180:~$ find ./ -name *.txt
```

```
./liczby.txt
```

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

`find` – wyszukiwanie plików/katalogów (`man find`)

```
student@wftlab-180:~$ find ./ -name *.txt  
./liczby.txt  
student@wftlab-180:~$
```

Proszę powtórzyć/opanować:

- **bash**: podstawowe polecenia, pętle i warunki (`if`, `test`) – pamiętać o poleceniu `man`,
- potok, `grep`, `find`,
- proste operacje na danych tekstowych (`cut`, `awk`, `sed`, `tr`) – ograniczone do omawianych.

## sed: edytor strumieniowy

Z dokumentacji **GNU sed** (GNU Free Documentation License / GNU GPL v3+):

*sed is a stream editor. A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline).\**

\*<https://www.gnu.org/software/sed/manual/sed.html#Introduction>



## sed: edytor strumieniowy

Z dokumentacji **GNU sed** (GNU Free Documentation License / GNU GPL v3+):

*sed is a stream editor. A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline).\**

\*<https://www.gnu.org/software/sed/manual/sed.html#Introduction>

Ważne: **substytucja wyrażen** – składnia:

```
sed 's/wyrazenie/zamiennik/'
```

## sed: edytor strumieniowy

Z dokumentacji **GNU sed** (GNU Free Documentation License / GNU GPL v3+):

*sed is a stream editor. A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline).\**

\*<https://www.gnu.org/software/sed/manual/sed.html#Introduction>

Ważne: **substytucja wyrażen** – składnia:

```
sed 's/wyrazenie/zamiennik/'
```

Przykład:

```
sed 's/pierwszy/drugi/'
```

## sed: edytor strumieniowy

Z dokumentacji **GNU sed** (GNU Free Documentation License / GNU GPL v3+):

*sed is a stream editor. A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline).\**

\*<https://www.gnu.org/software/sed/manual/sed.html#Introduction>

Ważne: **substytucja wyrażen** – składnia:

```
sed 's/wyrazenie/zamiennik/'
```

Przykład:

```
sed 's/pierwszy/drugi/'
```

```
sed 's/pierwszy/drugi/' plik.txt
```

Substytucja (s) domyślnie zastępuje wyłącznie pierwsze wystąpienie szukanego wyrażenia w każdej linii!

## sed: edytor strumieniowy

Substytucja (s) domyślnie zastępuje wyłącznie pierwsze wystąpienie szukanego wyrażenia w każdej linii!

Globalna substytucja (wszystkie wystąpienia we wszystkich liniach):

```
sed 's/pierwszy/drugi/g' plik.txt
```

## sed: edytor strumieniowy

Substytucja (s) domyślnie zastępuje wyłącznie pierwsze wystąpienie szukanego wyrażenia w każdej linii!

Globalna substytucja (wszystkie wystąpienia we wszystkich liniach):

```
sed 's/pierwszy/drugi/g' plik.txt
```

Znak ucieczki (ang. *escape character*): może nim być np. *backslash* (aby zacytować *slash*) lub podkreślnik (*\_*) jako separator.

## sed: edytor strumieniowy

Substytucja (s) domyślnie zastępuje wyłącznie pierwsze wystąpienie szukanego wyrażenia w każdej linii!

Globalna substytucja (wszystkie wystąpienia we wszystkich liniach):

```
sed 's/pierwszy/drugi/g' plik.txt
```

Znak ucieczki (ang. *escape character*): może nim być np. *backslash* (aby zacytować *slash*) lub podkreślnik (*\_*) jako separator.

```
sed 's_/pierwszy_/drugi_g' plik.txt
```

## **sed:** edytor strumieniowy

**&** – symbol odnoszący się do znalezionej wyrażenia (przydatny, jeżeli używamy wyrażenia regularnego i nie wiemy jak dokładnie wygląda).



## sed: edytor strumieniowy

**&** – symbol odnoszący się do znalezionej wyrażenia (przydatny, jeżeli używamy wyrażenia regularnego i nie wiemy jak dokładnie wygląda).

Przykład (przypadek trywialny):

```
sed 's/pierwszy/"pierwszy"/'
```

## sed: edytor strumieniowy

**&** – symbol odnoszący się do znalezionej wyrażenia (przydatny, jeżeli używamy wyrażenia regularnego i nie wiemy jak dokładnie wygląda).

Przykład (przypadek trywialny):

```
sed 's/pierwszy/"pierwszy"/'
```

(!) **ŹŁE:**

```
sed 's/[a-z]*/"pierwszy"/'
```

## sed: edytor strumieniowy

**&** – symbol odnoszący się do znalezionej wyrażenia (przydatny, jeżeli używamy wyrażenia regularnego i nie wiemy jak dokładnie wygląda).

Przykład (przypadek trywialny):

```
sed 's/pierwszy/"pierwszy"/'
```

(!) ŻŁE:

```
sed 's/[a-z]*/"pierwszy"/'
```

Można tak:

```
sed 's/[a-z]*/"&"/'
```

## sed: edytor strumieniowy

**&** – symbol odnoszący się do znalezionej wyrażenia (przydatny, jeżeli używamy wyrażenia regularnego i nie wiemy jak dokładnie wygląda).

Przykład (przypadek trywialny):

```
sed 's/pierwszy/"pierwszy"/'
```

(!) **ŹLE:**

```
sed 's/[a-z]*/"pierwszy"/'
```

Można tak:

```
sed 's/[a-z]*/"&"/'
```

Lub tak:

```
sed 's/[0-9]*/"&"/'
```

Inne możliwości:

- `[0-9]*` – zero lub więcej cyfr z przedziału 0-9,
- `[0-9][0-9]*` – jedna lub więcej cyfr z przedziału 0-9,
- `[0-9]+` – jak wyżej
- `[a-z]*` – zero lub więcej znaków,
- `[a-z][a-z]*` – jeden lub więcej znaków,
- `[a-z]+` – jak wyżej,
- `\1` – pierwsze znalezione (zapamiętane) wyrażenie,
- `\2` – drugie znalezione (zapamiętane) wyrażenie.

# sed: edytor strumieniowy

Inne możliwości:

- `[0-9]*` – zero lub więcej cyfr z przedziału 0-9,
- `[0-9][0-9]*` – jedna lub więcej cyfr z przedziału 0-9,
- `[0-9]+` – jak wyżej
- `[a-z]*` – zero lub więcej znaków,
- `[a-z][a-z]*` – jeden lub więcej znaków,
- `[a-z]+` – jak wyżej,
- `\1` – pierwsze znalezione (zapamiętane) wyrażenie,
- `\2` – drugie znalezione (zapamiętane) wyrażenie.

Przykład:

```
sed 's/\([a-z]+\)\ ([a-z]+)/\2 \1/'
```

Inne znaki specjalne i przykładowe wyrażenia:

- `^` – początek linii,
- `#` – pojedynczy znak,
- `$` – koniec linii,
- `^A` – znak "A" na początku linii,
- `A$` – znak "A" na końcu linii,
- `\n` – znak nowego wiersza.

## sed: edytor strumieniowy

Inne znaki specjalne i przykładowe wyrażenia:

- `^` – początek linii,
- `#` – pojedynczy znak,
- `$` – koniec linii,
- `^A` – znak "A" na początku linii,
- `A$` – znak "A" na końcu linii,
- `\n` – znak nowego wiersza.

Przykład:

```
sed 's/ /\n/' plik.txt
```



## sed: edytor strumieniowy

Inne znaki specjalne i przykładowe wyrażenia:

- `^` – początek linii,
- `#` – pojedynczy znak,
- `$` – koniec linii,
- `^A` – znak "A" na początku linii,
- `A$` – znak "A" na końcu linii,
- `\n` – znak nowego wiersza.

Przykład:

```
sed 's/ /\n/' plik.txt
```

```
sed 's/ /\n/g' plik.txt
```

## tr: zmień/usuń znaki

Przykład (-s, --squeeze-repeats):

```
$ echo "aaaaaaabbbbbbbccccccdddddde" | tr -s abcde
```

## tr: zmień/usuń znaki

Przykład (-s, --squeeze-repeats):

```
$ echo "aaaaaaabbbbbbbccccccdddddde" | tr -s abcde  
abcde  
$
```

## tr: zmień/usuń znaki

### Przykład (-s, --squeeze-repeats):

```
$ echo "aaaaaaabbbbbbbccccccdddddde" | tr -s abcde  
abcde  
$
```

### Przykład (-s, --squeeze-repeats):

```
$ echo "pierwszy      drugi      trzeci" | tr -s ' '
```

## tr: zmień/usuń znaki

Przykład (-s, --squeeze-repeats):

```
$ echo "aaaaaaabbbbbbbccccccdddddde" | tr -s abcde  
abcde  
$
```

Przykład (-s, --squeeze-repeats):

```
$ echo "pierwszy      drugi    trzeci" | tr -s ' '  
pierwszy drugi trzeci  
$
```

## tr: zmień/usuń znaki

### Przykład (-s, --squeeze-repeats):

```
$ echo "aaaaaaabbbbbbbccccccdddddde" | tr -s abcde  
abcde  
$
```

### Przykład (-s, --squeeze-repeats):

```
$ echo "pierwszy      drugi    trzeci" | tr -s ' '  
pierwszy drugi trzeci  
$
```

### Przykład (znajdź i zamień):

```
$ echo "To-jest-przykładowy-tekst" | tr '-' ' ' ,
```

## tr: zmień/usuń znaki

### Przykład (-s, --squeeze-repeats):

```
$ echo "aaaaaaabbbbbbbccccccdddddde" | tr -s abcde  
abcde  
$
```

### Przykład (-s, --squeeze-repeats):

```
$ echo "pierwszy      drugi    trzeci" | tr -s ' '  
pierwszy drugi trzeci  
$
```

### Przykład (znajdź i zamień):

```
$ echo "To-jest-przykładowy-tekst" | tr '-' ' '  
To jest przykładowy tekst  
$
```

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



```
student@wftlab-180:~$ cat dane.txt
```

```
student@wftlab-180:~$ cat dane.txt
```

```
# X   Y   Z
```

```
1  10  20
```

```
2  20  30
```

```
3  30  40
```

```
4  40  50
```

```
5  50  60
```

```
6  60  70
```

```
7  70  80
```

```
(...)
```

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

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

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

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

```
G N U P L O T
```

```
Version 5.0 patchlevel 3 last modified 2016-02-21
```

```
Copyright (C) 1986-1993, 1998, 2004, 2007-2016
```

```
Thomas Williams, Colin Kelley and many others
```

```
gnuplot home: http://www.gnuplot.info
```

```
faq, bugs, etc: type ‘help FAQ’
```

```
immediate help: type ‘help’ (plot window: hit ‘h’)
```

```
Terminal type set to ‘qt’
```

```
gnuplot>
```

# gnuplot – wykresy 2D

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

```
G N U P L O T
```

```
Version 5.0 patchlevel 3 last modified 2016-02-21
```

```
Copyright (C) 1986-1993, 1998, 2004, 2007-2016
```

```
Thomas Williams, Colin Kelley and many others
```

```
gnuplot home: http://www.gnuplot.info
```

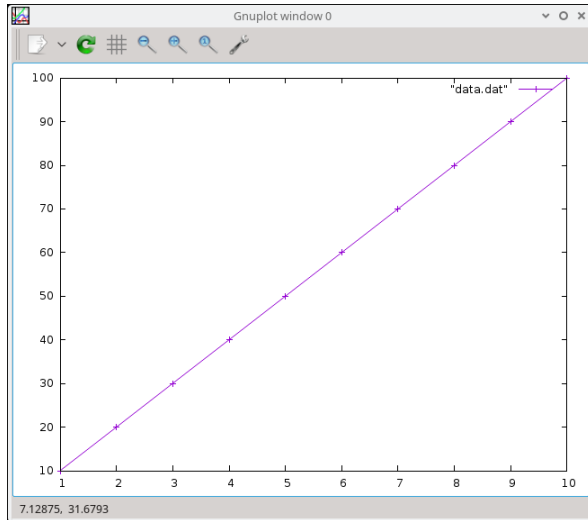
```
faq, bugs, etc: type ‘help FAQ’
```

```
immediate help: type ‘help’ (plot window: hit ‘h’)
```

```
Terminal type set to ‘qt’
```

```
gnuplot> plot ‘dane.txt’ with linespoints
```

# gnuplot – wykresy 2D



```
gnuplot>
```

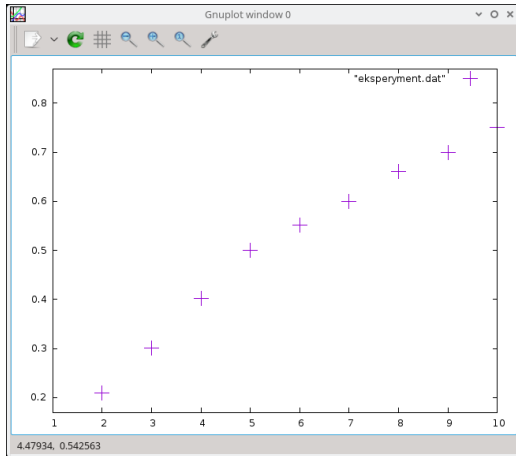


## gnuplot – wykresy 2D

```
gnuplot> plot 'eksperyment.dat' ps 3
```

# gnuplot – wykresy 2D

```
gnuplot> plot 'eksperyment.dat' ps 3
```



# gnuplot – wykresy 2D

```
gnuplot>
```

## gnuplot – wykresy 2D

```
gnuplot> f(x) =
```

## gnuplot – wykresy 2D

```
gnuplot> f(x) =a*x + b  
gnuplot>
```

## gnuplot – wykresy 2D

```
gnuplot> f(x) =a*x + b  
gnuplot> a=1  
gnuplot>
```

## gnuplot – wykresy 2D

```
gnuplot> f(x) =a*x + b  
gnuplot> a=1  
gnuplot> b=1  
gnuplot>
```

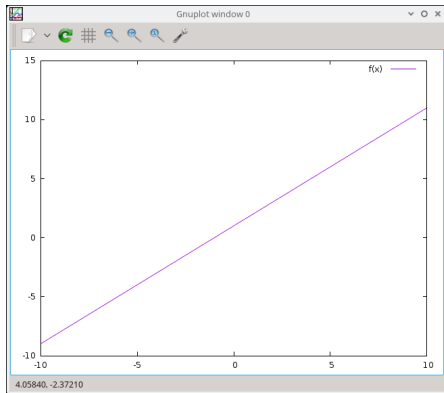
## gnuplot – wykresy 2D

```
gnuplot> f(x) =a*x + b  
gnuplot> a=1  
gnuplot> b=1  
gnuplot> plot f(x)
```



# gnuplot – wykresy 2D

```
gnuplot> f(x) =a*x + b  
gnuplot> a=1  
gnuplot> b=1  
gnuplot> plot f(x)
```



```
gnuplot>
```

```
gnuplot> fit f(x), 'eksperyment.dat' via a,b
```

```
gnuplot> fit f(x), 'eksperyment.dat' via a,b  
(...)
```

Final set of parameters Asymptotic Standard Error

a = 0.0710303 +/- 0.00438 (6.166%)

b = 0.0867333 +/- 0.02718 (31.33%)

(...)

```
gnuplot>
```

```
gnuplot> fit f(x), 'eksperyment.dat' via a,b  
(...)
```

Final set of parameters Asymptotic Standard Error

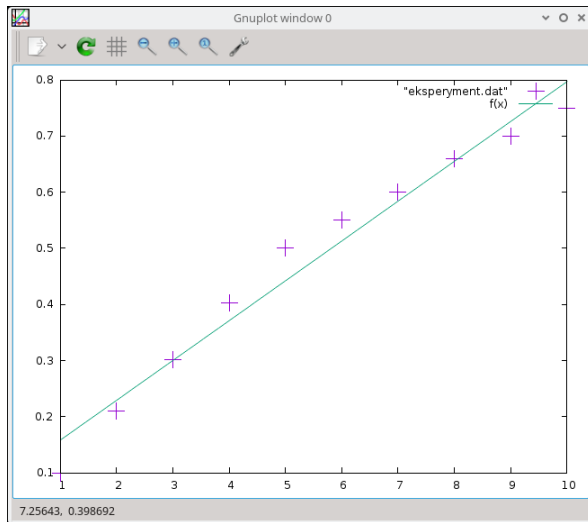
a = 0.0710303 +/- 0.00438 (6.166%)

b = 0.0867333 +/- 0.02718 (31.33%)

(...)

```
gnuplot> plot 'eksperyment.dat' ps 3, f(x)
```

# gnuplot – wykresy 2D – regresja liniowa



## Składnia:

```
gnuplot> plot [plik] [które kolumny] [jak]
```

### Składnia:

```
gnuplot> plot [plik] [które kolumny] [jak]
```

```
plot 'dane.txt' using 1:2 with linespoints pointsize 1 pointtype 7  
linewidth 2
```



### Składnia:

```
gnuplot> plot [plik] [które kolumny] [jak]
```

```
plot 'dane.txt' using 1:2 with linespoints pointsize 1 pointtype 7  
linewidth 2
```

```
plot 'dane.txt' u 1:2 w lp ps 1 pt 7 lw 2
```

## gnuplot – wykresy 2D – operacje na danych

### Składnia:

```
gnuplot> plot [plik] [które kolumny] [jak]
```

```
plot 'dane.txt' using 1:2 with linespoints pointsize 1 pointtype 7  
linewidth 2
```

```
plot 'dane.txt' u 1:2 w lp ps 1 pt 7 lw 2
```

```
plot 'dane.txt' using ($1+5):($2*2) w lp
```

# gnuplot – wykresy 2D – operacje na danych

## Składnia:

```
gnuplot> plot [plik] [które kolumny] [jak]
```

```
plot 'dane.txt' using 1:2 with linespoints pointsize 1 pointtype 7  
linewidth 2
```

```
plot 'dane.txt' u 1:2 w lp ps 1 pt 7 lw 2
```

```
plot 'dane.txt' using ($1+5):($2*2) w lp
```

```
plot 'dane.txt' u 1:2, 'dane.txt' u 1:3, 'dane.txt' u 1:4
```

- gnuplot – zapisywanie danych do pliku (`set terminal`, `set output`),

- gnuplot – zapisywanie danych do pliku (`set terminal`, `set output`),
- pliki konfiguracyjne powłoki (`$HOME/.bash_history`, `$HOME/.bash_profile`, `$HOME/.bashrc`),

- gnuplot – zapisywanie danych do pliku (`set terminal`, `set output`),
- pliki konfiguracyjne powłoki (`$HOME/.bash_history`, `$HOME/.bash_profile`, `$HOME/.bashrc`),
- Bash – `read`, zmienne specjalne/argumenty, `sort`, `tail`, `head`, `tac`,

- gnuplot – zapisywanie danych do pliku (`set terminal`, `set output`),
- pliki konfiguracyjne powłoki (`$HOME/.bash_history`, `$HOME/.bash_profile`, `$HOME/.bashrc`),
- Bash – `read`, zmienne specjalne/argumenty, `sort`, `tail`, `head`, `tac`,
- skrypt liczący silnię,

- gnuplot – zapisywanie danych do pliku (`set terminal`, `set output`),
- pliki konfiguracyjne powłoki (`$HOME/.bash_history`, `$HOME/.bash_profile`, `$HOME/.bashrc`),
- Bash – `read`, zmienne specjalne/argumenty, `sort`, `tail`, `head`, `tac`,
- skrypt liczący silnię,
- prosty kalkulator – cztery działania (nie tylko na liczbach całkowitych!),



- gnuplot – zapisywanie danych do pliku (`set terminal`, `set output`),
- pliki konfiguracyjne powłoki (`$HOME/.bash_history`, `$HOME/.bash_profile`, `$HOME/.bashrc`),
- Bash – `read`, zmienne specjalne/argumenty, `sort`, `tail`, `head`, `tac`,
- skrypt liczący silnię,
- prosty kalkulator – cztery działania (nie tylko na liczbach całkowitych!),
- definiowanie funkcji powłoki,

- gnuplot – zapisywanie danych do pliku (`set terminal`, `set output`),
- pliki konfiguracyjne powłoki (`$HOME/.bash_history`, `$HOME/.bash_profile`, `$HOME/.bashrc`),
- Bash – `read`, zmienne specjalne/argumenty, `sort`, `tail`, `head`, `tac`,
- skrypt liczący silnię,
- prosty kalkulator – cztery działania (nie tylko na liczbach całkowitych!),
- definiowanie funkcji powłoki,
- data/czas systemowy (ISO 8601).

# GNU/Linux: procesy

Proces to program działający (wykonujący zadania) w systemie operacyjnym (lista procesów – polecenie `ps`).

## Status procesu:

- R – *running* (działający),
- D lub S – uśpiony, ale (nie)możliwy do przerwania,
- Z – *zombie*,
- T – zatrzymany, (...)

Dodatkowo:

- < – wysoki priorytet,
- N – niski priorytet,
- 1 – wielowątkowy, (...)

## Procesy:

- są uruchamiane jako kopie procesu macierzystego,
- są identyfikowane m.in. przez PID, PPID oraz UID, (...),
- mogą się komunikować z użytkownikiem (`stdin`, `stdout`, `stderr`),
- po zakończeniu zwracają *kod zakończenia* (status),
- można je kontrolować za pomocą *sygnałów*,
- główny proces macierzysty: `init` (zależnie od implementacji),
- podlegają narzuconym ograniczeniom (`ulimit`),
- posiadają nadane priorytety,
- mogą (nie muszą) być związane z terminalem (te drugie to tzw. *daemony*),
- można je monitorować (np. narzędzia `ps`, `top`, `kill`, `xkill`),
- dane procesów znajdują się w systemie plików `/proc`.

Procesy użytkownika mogą działać w tle:

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

Procesy użytkownika mogą działać w tle:

```
student@wftlab-180:~$ ./skrypt.sh &
```

Procesy użytkownika mogą działać w tle:

```
student@wftlab-180:~$ ./skrypt.sh &  
student@wftlab-180:~$
```

Procesy użytkownika mogą działać w tle:

```
student@wftlab-180:~$ ./skrypt.sh &  
student@wftlab-180:~$
```

Uruchomione w tle procesy można wylistować poleceniem `jobs`, a wybrany proces podłączyć na powrót do terminala poleceniem `fg %id` (ang. *foreground*), gdzie `id` to numer zadania według `jobs`. Procesy podłączone do terminala można przerwać kombinacją `Ctrl+c` lub zatrzymać kombinacją `Ctrl+z`. Zatrzymane zadanie można być powtórnie wznowione w tle poleceniem `bg %id` (ang. *background*).



Inna możliwość – narzędzie screen:

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

Inna możliwość – narzędzie screen:

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

Inna możliwość – narzędzie screen:

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

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

## Inna możliwość – narzędzie screen:

```
student@wftlab-180:~$ screen  
student@wftlab-180:~$  
student@wftlab-180:~$ [Ctrl+A+D]
```

## Inna możliwość – narzędzie screen:

```
student@wftlab-180:~$ screen  
student@wftlab-180:~$  
student@wftlab-180:~$ [Ctrl+A+D]  
student@wftlab-180:~$
```

## Inna możliwość – narzędzie screen:

```
student@wftlab-180:~$ screen
student@wftlab-180:~$
student@wftlab-180:~$ [Ctrl+A+D]
student@wftlab-180:~$
student@wftlab-180:~$ screen -r
```

## Inna możliwość – narzędzie screen:

```
student@wftlab-180:~$ screen
student@wftlab-180:~$
student@wftlab-180:~$ [Ctrl+A+D]
student@wftlab-180:~$
student@wftlab-180:~$ screen -r
```

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

## Inna możliwość – narzędzie screen:

```
student@wftlab-180:~$ screen
student@wftlab-180:~$
student@wftlab-180:~$ [Ctrl+A+D]
student@wftlab-180:~$
student@wftlab-180:~$ screen -r
```

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



## Inna możliwość – narzędzie screen:

```
student@wftlab-180:~$ screen
student@wftlab-180:~$
student@wftlab-180:~$ [Ctrl+A+D]
student@wftlab-180:~$
student@wftlab-180:~$ screen -r
```

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

There are screens on:

```
3476.pts-5.hadron (19.11.2017 12:40:57) (Detached)
3469.pts-5.hadron (19.11.2017 12:40:54) (Detached)
3461.pts-5.hadron (19.11.2017 12:40:46) (Detached)
3440.pts-5.hadron (19.11.2017 12:39:03) (Detached)
4 Sockets in /run/screen/S-herman.
```

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

```
student@wftlab-180:~$ screen -r 3476.pts-5.hadron
```

```
student@wftlab-180:~$ screen -r 3476.pts-5.hadron  
student@wftlab-180:~$
```

```
student@wftlab-180:~$ screen -r 3476.pts-5.hadron  
student@wftlab-180:~$ screen -ls
```

```
student@wftlab-180:~$ screen -r 3476.pts-5.hadron
student@wftlab-180:~$ screen -ls
There are screens on:
3476.pts-5.hadron (19.11.2017 12:40:57) (Attached)
3469.pts-5.hadron (19.11.2017 12:40:54) (Detached)
3461.pts-5.hadron (19.11.2017 12:40:46) (Detached)
3440.pts-5.hadron (19.11.2017 12:39:03) (Detached)
4 Sockets in /run/screen/S-herman.
student@wftlab-180:~$
```

## Uruchamianie systemu (sysv):

BIOS/EFI → MBR → LILO/GRUB → KERNEL → INIT → RUN LEVEL(s)

## Uruchamianie systemu (sysv):

BIOS/EFI → MBR → LILO/GRUB → KERNEL → INIT → RUN LEVEL(s)

## RUN LEVELS (według **LSB**):

- 0 – zatrzymuje system,
- 1 – tryb *single-user* – administracja,
- 2 – tryb *multi-user*, bez obsługi sieci,
- 3 – tryb *normalny*,
- 4 – zależny od systemu/dystrybucji,
- 5 – taki sam jak 3, ale z graficznym menedżerem logowania,
- 6 – restart systemu.



## Procesy: *demony*

### Demon (ang. *daemon*):

proces **nie**interaktywny; nie jest połączony z terminalem. Daemonami są najczęściej programy realizujące usługi (sieciowe i nie tylko).

# Procesy: *demony*

## Demon (ang. *daemon*):

proces **nie**interaktywny; nie jest połączony z terminalem. Daemonami są najczęściej programy realizujące usługi (sieciowe i nie tylko).

## Usługi sieciowe:

- NNTP (*Network News Transfer Protocol*),
- SMTP/POP3/IMAP (poczta elektroniczna),
- HTTP (*Hyper Text Transfer Protocol*),
- FTP (*File Transfer Protocol*),
- SSH (*Secure SHell*), SSL (*Secure Socket Layer*), Telnet,
- PPP (*Point-to-Point Transfer Protocol*),
- (...)

## DNS: system nazw domenowych

rozproszona baza adresów sieciowych. Serwery DNS można *odpytywać* za pomocą narzędzia `whois`.

## DNS: system nazw domenowych

rozproszona baza adresów sieciowych. Serwery DNS można *odpytywać* za pomocą narzędzia `whois`.

## Architektura **klient-serwer**:

podział ról związany z usługami, które zapewnia serwer i z których korzystają klienci (oprogramowanie klienckie). Programy mogą też pracować w trybie P2P (*Peer-to-peer*).

## DNS: system nazw domenowych

rozproszona baza adresów sieciowych. Serwery DNS można *odpytywać* za pomocą narzędzia `whois`.

## Architektura **klient-serwer**:

podział ról związany z usługami, które zapewnia serwer i z których korzystają klienci (oprogramowanie klienckie). Programy mogą też pracować w trybie P2P (*Peer-to-peer*).

Jak sprawdzić co nasz system udostępnia *na zewnątrz*?

# Usługi sieciowe

## DNS: system nazw domenowych

rozproszona baza adresów sieciowych. Serwery DNS można *odpytywać* za pomocą narzędzia `whois`.

## Architektura **klient-serwer**:

podział ról związany z usługami, które zapewnia serwer i z których korzystają klienci (oprogramowanie klienckie). Programy mogą też pracować w trybie P2P (*Peer-to-peer*).

Jak sprawdzić co nasz system udostępnia *na zewnątrz*?

Na przykład skanując **porty** (program `nmap`).

# Usługi sieciowe

## DNS: system nazw domenowych

rozproszona baza adresów sieciowych. Serwery DNS można *odpytywać* za pomocą narzędzia `whois`.

## Architektura **klient-serwer**:

podział ról związany z usługami, które zapewnia serwer i z których korzystają klienci (oprogramowanie klienckie). Programy mogą też pracować w trybie P2P (*Peer-to-peer*).

Jak sprawdzić co nasz system udostępnia *na zewnątrz*?

Na przykład skanując **porty** (program `nmap`).

Oprogramowanie działające w systemie klient-serwer może pracować lokalnie (na przykład X Window System)!

# Usługi sieciowe

## DNS: system nazw domenowych

rozproszona baza adresów sieciowych. Serwery DNS można *odpytywać* za pomocą narzędzia `whois`.

## Architektura **klient-serwer**:

podział ról związany z usługami, które zapewnia serwer i z których korzystają klienci (oprogramowanie klienckie). Programy mogą też pracować w trybie P2P (*Peer-to-peer*).

Jak sprawdzić co nasz system udostępnia *na zewnątrz*?

Na przykład skanując **porty** (program `nmap`).

Oprogramowanie działające w systemie klient-serwer może pracować lokalnie (na przykład X Window System)!

Co z bezpieczeństwem sieciowym w GNU/Linuxie?



Napisać skrypt generujący statystykę systemową: ile procesów jest uruchomionych, a także jaka jest temperatura na mikroprocesorach. Dane przedstawić w formie wykresów.

**Czas na pytania i dyskusję**