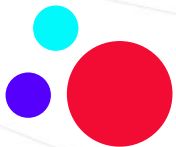


Podstawy Linuxa

infoShare Academy



HELLO

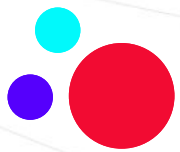
Maciej Małek

DevOps engineer
AWS, Linux SysAdmin, Terraform, Python



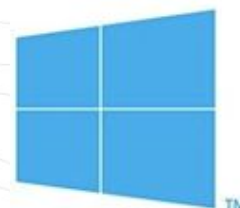
Czym jest system operacyjny?

Oprogramowanie zarządzające systemem komputerowym, tworzące środowisko do uruchamiania i kontroli zadań.



Współczesne systemy operacyjne

- Linux
- Windows
- macOS
- Android
- iOS
- i inne ...



Windows®



System operacyjny – zadania

- Planowanie i przydział czasu procesora dla zadań,
- Kontrola i przydział pamięci operacyjnej dla uruchomionych zadań,
- Dostarczanie mechanizmów do synchronizacji zadań i komunikacji pomiędzy zadaniami,
- Obsługa sprzętu,
- Ustalanie połączeń sieciowych,
- Zarządzanie plikami.



- Wolne i otwarte oprogramowanie
- Pierwsza wersja jądra Linuksa opublikowana 17 września 1991
- Twórca: fiński programista Linus Torvalds
- Geneza nazwy: Linus + Unix
- Potem też akronim rekurencyjny Linux Is Not Unix
- Aby zapoznać się, jak system ten działa od podstaw, można zbudować swoją wersję Linuksa, bazując na podręczniku Linux from scratch (LFS):

<https://www.linuxfromscratch.org/>



Jądro Linuxa (kernel)

- Największa część kodu napisana w C, pozostała część to wstawki w assemblerze.
- Obsługuje wielozadaniowość, wielowątkowość, pamięć wirtualną, biblioteki współdzielone, ładowanie na żądanie, współdzielony kod wykonywalny, zarządzanie pamięcią i obsługę sieci TCP/IP.

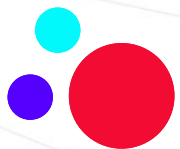


WSL <https://docs.microsoft.com/en-us/windows/wsl/install>

Putty (<https://www.putty.org/>)

GitBash <https://git-scm.com/download/win>

Windows Terminal <https://github.com/microsoft/terminal>

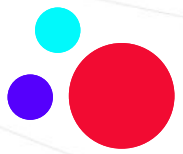


Jaka dystrybucja na stację roboczą?



elementary OS





Jaka dystrybucja na serwer?

RedHat



Fedora



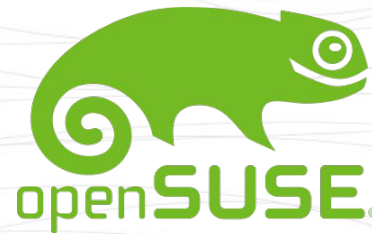
Ubuntu



Debian



SuSE



Amazon Linux



AlmaLinux



RockyLinux



Fedora CoreOS to automatycznie aktualizujący się, minimalistyczny, monolityczny system operacyjny skoncentrowany na kontenerach, zaprojektowany dla klastrów, ale także działający samodzielnie, zoptymalizowany pod kątem Kubernetes, ale także świetny bez niego.





Fedora vs Fedora CoreOS

- Dostosowywany podczas instalacji poprzez plik konfiguracyjny (ignition)
- Ignition file - może partycjonować dyski, formatować system plików, tworzyć użytkowników, zapisywać pliki w tym usługi systemd.
- System (immutable) - wszystko jest tylko do odczytu poza katalogami /etc i /var które są dostępne do odczytu i zapisu.
- Infrastruktura powinna być zamknięta w pliku konfiguracyjnym Ignition i kontenerach.



Na potrzeby kursu potrzebujemy Ubuntu

Vagrant

```
vagrant init hashicorp/bionic64
```

```
vagrant up
```

```
vagrant ssh
```

```
vagrant destroy
```

Może być też „po prostu” Ubuntu, VirtualBox (bez Vagranta), WSL (Windows Subsystem for Linux), AWS EC2, etc.



Podstawy Linuxa

Podstawowe informacje.



System plików

- **ext**
- **ext2** – ulepszony ext
- **ext3** – bazujący na ext2, z księgowaniem oraz szybszymi operacjami na katalogach
- **ext4** – bazujący na ext3, z lepszą alokacją miejsca na dysku, zwiększonymi limitami, możliwością defragmentacji online (Ubuntu)
- **xfs** – system plików przygotowany z myślą o dużej wydajności, mniej podatny na defragmentację niż ext4 (RedHat)
- **btrfs** – nowoczesny system plików z księgowaniem, migawkami, klonowaniem, zarządzaniem wolumenami wzorowany na ZFS (Fedora, SuSE)
- **swap** – w zasadzie brak systemu plików, przestrzeń wymiany (struktura)



Najważniejsze katalogi

/bin binarne (wykonywalne) pliki najbardziej podstawowych narzędzi systemowych

/boot pliki niezbędne do uruchomienia systemu (kernel, initrd, pliki bootloadera – w przypadku GRUB)

/dev znajdujące się tutaj pliki nie są faktycznie plikami na dysku, lecz odnoszą się do urządzeń – za ich pośrednictwem system komunikuje się z urządzeniami (komunikacja niskopoziomowa)

/etc pliki konfiguracyjne, ustawienia systemowe

/home zwyczajowe miejsce na katalogi domowe użytkowników



Najważniejsze katalogi

/lib systemowe biblioteki dzielone (shared libraries), zawierające funkcje które są wykonywane przez wiele różnych programów

/media punkt montowania nośników wymiennych

/mnt punkt tymczasowego montowania systemów plików

/proc wirtualny katalog, zawierający dane o aktualnie uruchomionych procesach

/root katalog domowy użytkownika root – głównego administratora każdego systemu uniksowego, który ma maksymalne uprawnienia



Najważniejsze katalogi

/sbin pliki wykonywalne aplikacji systemowych

/tmp pliki tymczasowe

/usr dodatkowe programy, które umożliwiają pracę zwykłemu użytkownikowi systemu

/var pliki systemowe, ale których zawartość często się zmienia, jak logi programów/systemu, pliki html czy skrypty php/cgi wykorzystywane przez serwer www –
inaczej mówiąc są to dane zapisywane przez system i ważniejsze programy

- W Linuksie wszystko jest plikiem (standardowe pliki, urządzenia, połączenia sieciowe)
- Każdy plik (bezpośrednio lub pośrednio) wskazuje na i-node
 - inode'y są strukturami opisującymi pliki w systemie – zawierają wszelkie informacje związane z plikiem z wyłączeniem danych pliku oraz jego nazwy
 - inode nie zawiera nazwy pliku, ponieważ wiele plików (o różnych nazwach) może wskazywać na ten sam inode
 - inode jest unikalny w obrębie jednego systemu plików
 - kopiowanie tworzy nowy inode, przenoszenie – nie, zmienia tylko zawartość katalogu
- Podgląd informacji:

```
~$ stat plik_testowy.txt

File: plik_testowy.txt

Size: 19          Blocks: 8          IO Block: 4096   regular file

Device: 850h/2128d    Inode: 29523       Links: 1

Access: (0644/-rw-r--r--)  Uid: ( 1000/   maciej)   Gid: ( 1000/   maciej)

Access: 2023-01-08 10:52:50.130777534 +0100
Modify: 2023-01-08 10:53:00.650830258 +0100
Change: 2023-01-08 10:53:00.650830258 +0100
Birth: 2023-01-08 10:52:50.130777534 +0100
```



<https://github.com/ohmyzsh/ohmyzsh>

```
sh -c "$(curl -fsSL https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"
```

```
Using the Oh My Zsh template file and adding it to ~/.zshrc.

Time to change your default shell to zsh:
Do you want to change your default shell to zsh? [Y/n] Y
Changing your shell to /usr/bin/zsh...
Shell successfully changed to '/usr/bin/zsh'.

ohmyzsh ....is now installed!

Before you scream Oh My Zsh! look over the '.zshrc' file to select plugins, themes, and options.

• Follow us on Twitter: https://twitter.com/ohmyzsh
• Join our Discord community: https://discord.gg/ohmyzsh
• Get stickers, t-shirts, coffee mugs and more: https://shop.planetargon.com/collections/oh-my-zsh

+ .oh-my-zsh git:(master)
+ .oh-my-zsh git:(master) ls -la
total 120
drwxr-xr-x 12 root root 4096 Feb 11 19:33 .
drwx----- 3 root root 4096 Feb 11 19:34 ..
drwxr-xr-x 3 root root 4096 Feb 11 19:34 cache
-rw-r--r-- 1 root root 3374 Feb 11 19:33 CODE_OF_CONDUCT.md
-rw-r--r-- 1 root root 8281 Feb 11 19:33 CONTRIBUTING.md
drwxr-xr-x 4 root root 4096 Feb 11 19:33 custom
-rw-r--r-- 1 root root 115 Feb 11 19:33 .editorconfig
drwxr-xr-x 8 root root 4096 Feb 11 19:33 .git
drwxr-xr-x 5 root root 4096 Feb 11 19:33 .github
-rw-r--r-- 1 root root 77 Feb 11 19:33 .gitignore
-rw-r--r-- 1 root root 131 Feb 11 19:33 .gitpod.Dockerfile
-rw-r--r-- 1 root root 259 Feb 11 19:33 .gitpod.yml
drwxr-xr-x 2 root root 4096 Feb 11 19:33 lib
-rw-r--r-- 1 root root 1142 Feb 11 19:33 LICENSE.txt
drwxr-xr-x 2 root root 4096 Feb 11 19:34 log
-rw-r--r-- 1 root root 5927 Feb 11 19:33 oh-my-zsh.sh
drwxr-xr-x 313 root root 12288 Feb 11 19:33 plugins
-rw-r--r-- 1 root root 13840 Feb 11 19:33 README.md
-rw-r--r-- 1 root root 1083 Feb 11 19:33 SECURITY.md
drwxr-xr-x 2 root root 4096 Feb 11 19:33 templates
drwxr-xr-x 2 root root 4096 Feb 11 19:33 themes
drwxr-xr-x 2 root root 4096 Feb 11 19:33 tools
+ .oh-my-zsh git:(master) |
```

~ – katalog domowy

.. – katalog nadrzędny

. – katalog bieżący

| – przekierowanie wyjścia standardowego (pipe)

> – nadpisuje dane w pliku

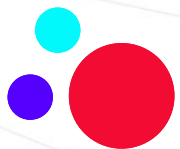
>> – dołącza dane na koniec pliku

&& – “and” – wykonaj następną komendę tylko jeżeli poprzednia zakończy się powodzeniem

|| – “or” – wykonaj następną komendę tylko jeżeli poprzednia zakończy się błędem



Zmienne specjale i exit codes



Polecenia do operowania na katalogach

- **cd** - – zmienia katalog na poprzedni
- **cd** ~ – przechodzi do katalogu domowego
- **cd** .. – przechodzi do katalogu nadrzędnego
- **cd** /tmp – przechodzi do wskazanego katalogu
- **pwd** – pokazuje aktualny katalog
- **ls** – listuje zawartość katalogu (domyślnie bieżącego)
- **ls** -A – wyświetl prawie wszystko (także pliki ukryte ale bez . i ..)
- **ls** -l – używa długiego formatu (long listing)
- **ls** -i – pokazuje numer inode'a
- **ls** -Ali ~ – wszystko na raz (w katalogu domowym użytkownika)



Kopiowanie i przenoszenie

cp – komenda do kopiowania

- **cp ~/plik.txt .** – kopiuje plik.txt z katalogu domowego do bieżącego katalogu
- **cp -r katalog-testowy /tmp** – kopiuje katalog katalog-testowy” (z zawartością) do /tmp
- **cp test.txt wazny-plik.txt** – kopiuje plik test.txt z bieżącego katalogu do niego samego, ale zmieniając nazwę

mv – komenda do przenoszenia plików

- **mv ~/plik.txt nowa-nazwa.txt** – przenosi plik.txt z katalogu domowego do bieżącego katalogu, zmieniając jednocześnie nazwę pliku



Menadżer pakietów

- Pakiet – skompilowane źródło programu
- Większość dostępna w standardowych źródłach danej dystrybucji
- Instalacja:
 - Linuxy „typu Debianowego” (np. Ubuntu) **apt / apt-get** (Advanced Packaging Tool)
 - Linuxy „typu Red-Hatowego” **yum / dnf** (Yellowdog Updater, Modified / Dandified YUM)
- Przykłady:
 - `sudo apt install git` – instalacja git’a
 - `sudo apt update` – aktualizacja informacji o pakietach ze skonfigurowanych źródeł
 - `sudo apt list --upgradable` – wyświetla listę pakietów które mogą być zaktualizowane
 - `sudo apt update` – aktualizacja wszystkich pakietów które mają dostępne nowe wersje
 - `sudo apt install --only-upgrade package_name` – aktualizacja wybranego pakietu



Zarządzanie pakietami

- `apt remove <pakiet>` – usuwa pakiet
- `apt get purge <pakiet>` – usuwa pakiet i pliki konfiguracyjne z nim
- Combo

`sudo apt update && sudo apt upgrade`

`apt-get update && apt-get dist-upgrade && apt-get autoremove`

- `/etc/apt/sources.list` zawiera listę URLi do źródeł pakietów



Zarządzanie pakietami – dodawanie repo

Najnowszy Docker:

Przygotowanie (pakiety)

```
sudo apt-get remove docker docker.io containerd runc
```

```
sudo apt install ca-certificates curl gnupg lsb-release
```

Przygotowanie (klucz GPG)

```
sudo mkdir -p /etc/apt/keyrings
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o  
/etc/apt/keyrings/docker.gpg
```

```
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

Pobieramy i dodajemy oficjalny klucz GPG

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]  
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee  
/etc/apt/sources.list.d/docker.list > /dev/null
```


```
sudo apt update
```

```
sudo apt install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```




Edytory tekstu

- mcedit
- nano
- vim



```
automake [----] 0 L:[ 97+14 111/8017] *(3116/216033b)= p 112 0x70
{
  &{($self->_finish) ()};
}

sub target_hook ($$$$)
{
  my ($self) = @_;
  if (defined $self->_target_hook)
  {
    &{($self->_target_hook) ($)};
  }
}

package Automake;

require 5.005;
use strict 'vars', 'subs';
use File::Basename;
use IO::File;

my $ac = basename ($0);

## ----- ##
## Constants. ##
## ----- ##

# Parameters set by configure. Not to be changed. NOTE: assign
# VERSION as string so that eg version 0.30 will print correctly.
my $VERSION = "1.5.1a";
my $PACKAGE = "automake";
my $prefix = "/usr/local";

1)help 2)save 3)mark 4)replac 5)copy 6)move 7)search 8)delete 9)pullDn 10)quit
```



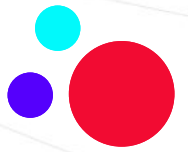

Przydatne narzędzia

- **less** / **more** / **cat** / **tac** – czytają pliki
 - **head** / **tail** – także czyta plik, ale tylko początek/koniec (domyślnie 10 linijek)
 - **mc** – Midnight Commander – menadżer plików
 - **grep** – wyszukuje w tekście linie zawierające ciąg znaków pasujący do podanego wyrażenia regularnego
-
- Np. `grep 'Ala' nazwa_pliku`



Przydatne narzędzia - sieć

- **ip** – konfiguracja sieci
- **curl** – narzędzie ogólnego przeznaczenia do przesyłania danych z/do serwera
- **wget** – program do pobierania sieciowego
- **ping** – wysyła 'pinga' do hosta (protokół ICMP)
- **awk** – język wyszukiwania i operowania na wzorcach
- **sed** – edytor strumieni tekstu: filtrowanie i modyfikowanie
- **sed** 's/unix/linux/' plik.txt
- **nmap** (**nping**, **nmap**)



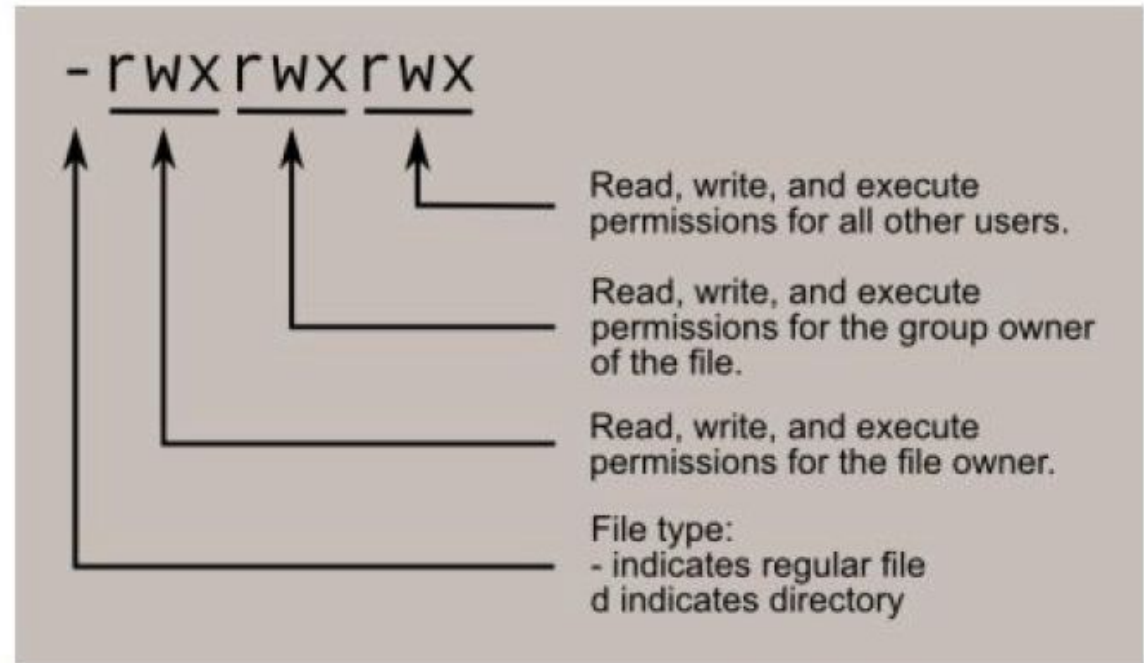
Polecenia do operowania na katalogach i plikach

- **mkdir** – tworzy katalog
- **touch** – tworzy plik
- **rm** – usuwa plik(i) / katalog(i)
- **rm -r** – usuwa rekursywnie (łącznie z katalogami)
- **rm -rf** – usuwa rekursywnie i nie pyta
- Uwaga na **rm -rf /**

- **super user do**
 - Z wielką władzą wiąże się wielka odpowiedzialność
 - Operacje wykonywane z uprawnieniami innego użytkownika, zwykle administratora („roota”)
-
- np.
 - `sudo ls`
 - `sudo su -`

Uprawnienia

- **r** – odczyt (read - 4)
 - **w** – zapis (write - 2)
 - **x** – wykonanie (execution - 1)
-
- **u** – właściciel (user)
 - **g** – grupa (group)
 - **o** – pozostali (other)
-
- `chmod g+w` – doda grupie prawo zapisu
 - `chmod o-rx` – zabierze uprawnienie odczytu i wykonania dla pozostałych



źródło: <https://linuxcommand.org/>



Uprawnienia liczbowo

- Potęgi dwójki:
- $r = 4$
- $w = 2$
- $x = 1$
- `chmod 700` – da wszystkie uprawnienia właścicielowi i zabierze wszystkie grupie i pozostałym
- `chmod 754 app.sh` – `rwX r-X r--`
- `chmod 750 script.sh`

$u = \text{rwX}, g = \text{r-X}, o = \text{---}$

	u g o		
	754		
access	r w x	r w x	r w x
binary	4 2 1	4 2 1	4 2 1
enabled	1 1 1	1 0 1	1 0 0
result	4 2 1	4 0 1	4 0 0
total	7	5	4

Źródło: <https://danielmiessler.com/>



Uprawnienia – po co? Kiedy?

- Zasada najmniejszego uprzywilejowania (Principle of least privilege)
- Nadajemy dostęp:
 - Nowym użytkownikom (także tym po ssh)
 - Serwisom

Linki symboliczne:

- `ln -s plik nazwa_linku`

Link twarde:

```
ln plik nazwa_linku
```

Przykład:

```
$ ls -li
```

```
total 8
```

```
29523 -rwxr-x--- 2 maciej maciej 19 Jan  8 10:53 hard-link-do-pliku
```

```
29523 -rwxr-x--- 2 maciej maciej 19 Jan  8 10:53 plik.txt
```

```
955 lrwxrwxrwx 1 maciej maciej  8 Jan  8 16:41 soft-link-do-pliku -> plik.txt
```




Przydatne narzędzia – archiwizacja

- tar / gzip / zip / gunzip – archiwizacja i/lub kompresowanie
- Spakowanie
 - .tar.gz: `tar -cvzf wynik.tar.gz plik_lub_katalog`
 - .zip: `zip wynik.zip plik_lub_katalog`
- Rozpakowanie
 - .tar.gz: `tar -xvzf nazwa_pliku.tar.gz`
 - .gz: `gunzip nazwa_pliku.gz`
 - .zip: `unzip nazwa_pliku.zip`




Metryki systemowe

- `top` – podstawowe źródło metryk
- `htop`
- `lscpu` – informacje o procesorze
- `free -h` – informacje o pamięci

<https://goinbigdata.com/understanding-output-of-free-in-ubuntu-16-04/>

- `df` – informacje o zamontowanych systemach plików
- `du` – informacje o użyciu dysku
- `ss -tulpn` – wyświetla połączenia sieciowe, statystyki, etc.
- `netstat -tulpn` – j/w (starsze)



Metryki systemowe – load average

- Np. z uptime: 23:16:49 up 10:49, 5 user, load average: 1.00, 0.40, 3.35
- Dla systemu 1-procesorowego:
 - średnio 100% użycia w ostatniej minucie
 - średnio 40% zużycia (60% „idle”) w ostatnich 5 minutach
 - średnio 235% nadmiarowego obciążenia („overload”) w ostatnich 15 minutach; średnio 2.35 procesu czekało na swój przydział CPU





init jest pierwszym procesem uruchomionym po starcie systemu

Otrzymuje identyfikator procesu (PID) 1

Uruchamia on inne usługi startujące w tle, nowsze systemy init pozwalają również w wygodny sposób zarządzać zależnościami usług

Popularne systemy init

- **SysVinit:**

- Przed upowszechnieniem się systemd, najpopularniejszy system init w dystrybucjach Linux
- Polega na istnieniu siedmiu (0-6) poziomów uruchomienia (runlevel)
- Uruchamia skrypty sekwencyjnie, jeden po drugim

- **Upstart**

- Po raz pierwszy wprowadzony w Ubuntu, na krótko przyjęty w innych dystrybucjach, ostatecznie wyparty przez systemd
- Pozwala uruchamiać usługi równolegle
- Umożliwia śledzenie stanu usług, dzięki czemu możliwe jest automatyczne uruchomienie usługi, która uległa awarii

- **Systemd**

- Obecnie domyślny system init w głównych dystrybucjach
- Działanie opiera się na "jednostkach" (unit)

Domyślny typ unitu to service. W przypadku większości poleceń, jeśli pominiemy typ, właśnie ten zostanie użyty

systemctl bez żadnych opcji, lub **systemctl list-unit** – wyświetli listę wszystkich uruchomionych unitów

systemctl status <nazwa_unitu> – wyświetli informacje o stanie unitu (czy jest uruchomiony, od kiedy itp) w formacie wygodnym dla człowieka. Oprócz tego wyświetli ostatnie wpisy logów dotyczących danego unitu (jednak w tym przypadku wymagane są odpowiednie uprawnienia)

systemctl start <nazwa_unitu> – uruchamia wskazany unit

systemctl stop <nazwa_unitu> – zatrzymuje wskazany unit

systemctl restart <nazwa_unitu> – restartuje unit. Jeśli wcześniej był zatrzymany, wystartuje go

systemctl reload <nazwa_unitu> – próbuje odczytać i wdrożyć konfigurację specyficzną dla aplikacji. Nie należy mylić tego z ponownym odczytaniem konfiguracji unitu



Tworzenie własnego unitu

Definicje unitów są plikami tekstowymi, zawierającymi pary klucz-wartość, podzielone na sekcje. Definicja unitu typu service składa się z następujących sekcji :

- **[Unit]**: zawiera podstawowe informacje o samym unicie oraz jego zależności
- **[Service]**: zawiera parametry samej usługi, definiując sposób, w jaki jest ona zarządzana
- **[Install]**: zawiera parametry wymagane do instalacji unitu.

Zawartość jest używana przez komendy enable oraz disable

Jest wiele miejsc w systemie plików, gdzie można umieścić definicje unitów. Mogą się one różnić w zależności od dystrybucji. Listę przeszukiwanych lokalizacji wraz z opisem znajdziemy w podręczniku man systemd.unit. Można założyć, że bezpiecznymi miejscami są

`/etc/systemd/system` - dla unitów systemowych

`/etc/systemd/user` - dla unitów użytkownika



Service unit – [Unit]

Opis zachowania, zależności

[Unit]

Description=The Apache HTTP Server

After=network.target httpd-init.service

Requires=avahi-daemon.socket

Description – opis usługi w postaci czytelnej dla człowieka

After oraz **Before** – wskazują kolejność w jakiej usługi będą uruchamiane oraz zatrzymywane

Wants – wymusza uruchomienie innych usług wskazanych w tym parametrze. Nie określa jednak kolejności ich uruchomienia. Jest to zależność słaba, wobec czego, w przypadku gdy nie uda się uruchomić niektórych z zależnych usług, nie będzie to miało wpływu na start tej usługi

Requires – działa podobnie do Wants, jest jednak silną zależnością. Jeśli uruchomienie którejś ze znajdujących się tu usług nie powiedzie się, a jednocześnie zależna usługa przypisana została do parametru After, usługa którą opisuje ten unit również nie uruchomi się. Niezależnie od tego, jeśli któraś ze znajdujących się tu usług zostanie zatrzymana lub uruchomiona ponownie, ta sama akcja wykonana zostanie na tym unicie



Service unit – [Service]

Co ma wydarzyć się po wystartowaniu serwisu

```
[Service]
```

```
ExecStart=/usr/sbin/NetworkManager --no-daemon TimeoutSec=300
```

ExecStart – zawiera polecenie, które ma zostać uruchomione w celu uruchomienia usługi

ExecStop – definiuje polecenie służące do zatrzymania usługi.

Type – typ uruchomionej usługi, opisujący jej zachowanie. Najważniejsze typy:

- **simple** – domyślny typ w przypadku, gdy nie wskazano innego oraz zdefiniowano opcję ExecStart (oraz nie ustawiono opcji BusName). Usługa uważana jest za uruchomioną od razu po wywołaniu głównego polecenia
- **forking** – typ przeznaczony dla klasycznych usług systemu Linux, gdzie główny proces wywołuje w tle proces potomny i kończy swoje działanie. W tym przypadku usługa uważana jest za uruchomioną po zakończeniu działania przez główny proces
- **notify** – typem tym oznacza się usługi obsługujące protokół powiadomień systemd. Usługa tego typu uznawana jest za uruchomioną, kiedy systemd otrzyma powiadomienie
- **oneshot** – typ używany głównie do usług, które mają wykonać jakieś zadanie i zakończyć swoją pracę. Systemd uznaje, że usługa zakończyła uruchamianie, kiedy główne polecenie się zakończy. Ten typ jako jedyny zezwala na zdefiniowanie wielokrotnie opcji ExecStart lub ExecStop

User – określa użytkownika, w imieniu którego usługa zostanie uruchomiona

Restart – określa czy usługa powinna zostać zrestartowana po zatrzymaniu, a jeśli tak, to w jakich sytuacjach. Może przyjąć jedną z następujących wartości: no, on-success, on-failure, on-abnormal, on-watchdog, on-abort, always



Service unit – [Install]

Co ma wydarzyć się przy instalacji serwisu

```
[Install]
```

```
WantedBy=multi-user.target
```

WantedBy oraz **RequiredBy** – podobnie jak analogiczne opcje w sekcji Unit, definiują unity, od których powinna zależeć ta usługa. Pliki unitów, na które wskazują te opcje nie są modyfikowane. Zamiast tego tworzone są katalogi `.wants/` lub `.requires/` w których znajdują się dowiązania do zależnych unitów

Also – wskazuje dodatkowe unity, które powinny być zainstalowane (lub usunięte) wraz z tą usługą



Service unit – instalacja

`/usr/lib/systemd/system` – unity systemowe

`/etc/systemd/system` – unity użytkownika

Start: `systemctl start nasz_serwis.service`

Stop: `systemctl stop nasz_serwis.service`

Weryfikacja aktywności: `systemctl is-active nasz_serwis.service`



Service unit – autostart

`systemctl enable <nazwa_unit>` – aktywuje wybrany unit tak, aby był on uruchamiany

automatycznie (uruchomienie nastąpi przy kolejnym starcie systemu, aby uruchomić go natychmiast, należy dodać przełącznik `--now`)

`systemctl disable <nazwa_unit>` – dezaktywuje wybrany unit tak, aby nie był on dłużej

uruchamiany automatycznie (uruchomienie nastąpi przy kolejnym starcie systemu, aby uruchomić go natychmiast, należy dodać przełącznik `--now`)

`systemctl daemon-reload <nazwa_unit>` – wczytuje ponownie konfigurację unitów. Należy wywołać to polecenie, jeśli zmieniliśmy plik unitu.



service unit – journal

Komponent systemd odpowiedzialny za logi ze wszystkich źródeł

Tworzy plik binarny

`journalctl` – podgląd logów

`journalctl -b` – podgląd logów (tylko obecne uruchomienie)

`journalctl -u nasz_serwis.service` – podgląd logów naszego serwisu



service unit – przekierowanie logów

Sekcja

```
[Service]
```

```
StandardOutput=file:/var/log/nasz_serwis/log.log
```

```
StandardError=file:/var/log/nasz_serwis/log.err
```





- git init – inicjuje nowe repozytorium
- git clone – klonuje („pobiera”) repozytorium
- git checkout branch_name – zmiana brancha, jeżeli nie istnieje to „-b”
- git add – dodaje pliki do zmiany
- git commit – „komituje” zmiany
- git push – wypycha zmiany do zdalnego repozytorium (origina)
- git pull – pobiera zmiany ze zdalnego repozytorium (origina)
- git status – sprawdzanie stanu
- git tag -a v.0.1 -m „Pierwsza wersja” – tagowanie commit’u

<https://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository>





Pierwszy skrypt

```
#!/bin/bash
```

```
echo "Hello World"
```

Warto pamiętać o dodaniu praw do wykonywania (+x).

```
bash nazwa_skryptu.sh
```

```
./nazwa_skryptu.sh
```

echo \$? - sprawdzenie exit code'u ostatniej komendy (0 - Success, 1 - Fail)

```
#!/bin/bash
```

```
#Dodaj liczby  
((sum=25+35))
```

```
#Wypisz wynik  
echo ${sum}
```



Operacje arytmetyczne

subtraction

```
[me@linux ~]$ expr 1 - 1
```

0

addition

```
[me@linux ~]$ expr 1 + 1
```

2

assign result to a variable

```
[me@linux ~]$ myvar=$(expr 1 + 1)
```

```
[me@linux ~]$ echo $myvar
```

2

addition with a variable

```
[me@linux ~]$ expr $myvar + 1
```

3

division

```
[me@linux ~]$ expr $myvar / 3
```

0

multiplication

```
[me@linux ~]$ expr $myvar \* 3
```

6



```
#!/bin/bash
for (( counter=10; counter>0; counter-- ))
do
    echo -n "${counter}"
done
printf "\n"
```



```
#!/bin/bash
```

```
valid=true
```

```
count=1
```

```
while [ ${valid} ]
```

```
do
```

```
    echo ${count}
```

```
    if [ ${count} -eq 5 ]; then
```

```
        break
```

```
    fi
```

```
    ((count++))
```

```
done
```




Instrukcja if / else

```
#!/bin/bash
```

```
n=10
```

```
if [ $n -lt 10 ]; then
```

```
    echo "To liczba jednocyfrowa"
```

```
elif [ $n -lt 100 ]
```

```
then
```

```
    echo "To liczba dwucyfrowa"
```

```
else
```

```
    echo "To liczba trzy lub więcej cyfrowa"
```

```
fi
```



Wprowadzanie danych przez użytkownika

```
#!/bin/bash
```

```
echo "Podaj imię"
```

```
read name
```

```
echo "Czesc $name!"
```



Zmienne specjalne

`${0}` – nazwa bieżącego skryptu lub powłoki wraz ze ścieżką

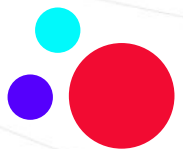
`${1} .. ${9}` – parametry przekazywane do skryptu

`${#}` – Zawiera liczbę argumentów przekazanych do funkcji

`${@}` – wszystkie parametry przekazywane do skryptu, równoważne `$1 $2 $3...`, jeśli nie podane są żadne parametry `$@` interpretowana jest jako nic.

`${?}` – kod powrotu ostatnio wykonywanego polecenia (0 jeśli polecenie wykonało się poprawnie, inna wartość jeśli błędnie).

`$$` – PID procesu bieżącej powłoki



Zmienne specjalne – przykład

```
#!/bin/bash
echo "Pełna nazwa: ${0}"
echo "Tylko nazwa skryptu: ${0##*/}"
echo "Ilość parametrów: ${#}"
echo "Wszystkie parametry: ${@}"
n=1
for param in ${@}
do
    echo "Parametr numer ${n} = ${param}"
    ((n++))
done
echo "Pid skryptu to: $$"
```

```
maciej@MACIEK-PROBOOK:~$ /tmp/katalog-testowy/script.sh param1 param2 param3 param4 param5
Pełna nazwa: /tmp/katalog-testowy/script.sh
Tylko nazwa skryptu: script.sh
Ilość parametrów: 5
Wszystkie parametry: param1 param2 param3 param4 param5
Parametr numer 1 = param1
Parametr numer 2 = param2
Parametr numer 3 = param3
Parametr numer 4 = param4
Parametr numer 5 = param5
Pid skryptu to: 9009
```

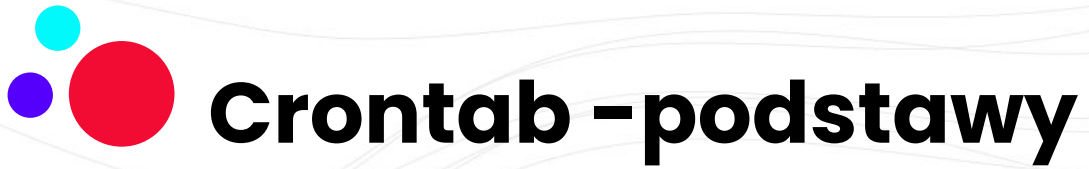


```
#!/bin/bash
```

```
pole_prostokata() {  
    area=$(( $1 * $2 ))  
    echo "Pole to $area"  
}
```

```
pole_prostokata 10 20
```





Crontab -podstawy

```
* * * * * [UserName] Command_to_execute
```

```
- - - - -
```

```
| | | | |
```

```
| | | | +-- Day of week (0-7) (Sunday=0 or 7) or Sun, Mon, Tue,...
```

```
| | | +---- Month (1-12) or Jan, Feb,...
```

```
| | +----- Day of month (1-31)
```

```
| +----- Hour (0-23)
```

```
+----- Minute (0-59)
```

```
SHELL=/bin/bash
```

```
MAILTO=admin@gmail.com
```

```
CRON_TZ=Europe/Warsaw
```

```
crontab -l
```

```
crontab -e
```

Dziękuję za uwagę!

infoShareAcademy.com