



# LINUX



# A konkretniej: serwery

projekt stworzony w ramach Stypendiów Śląskich

szymon pokrywka

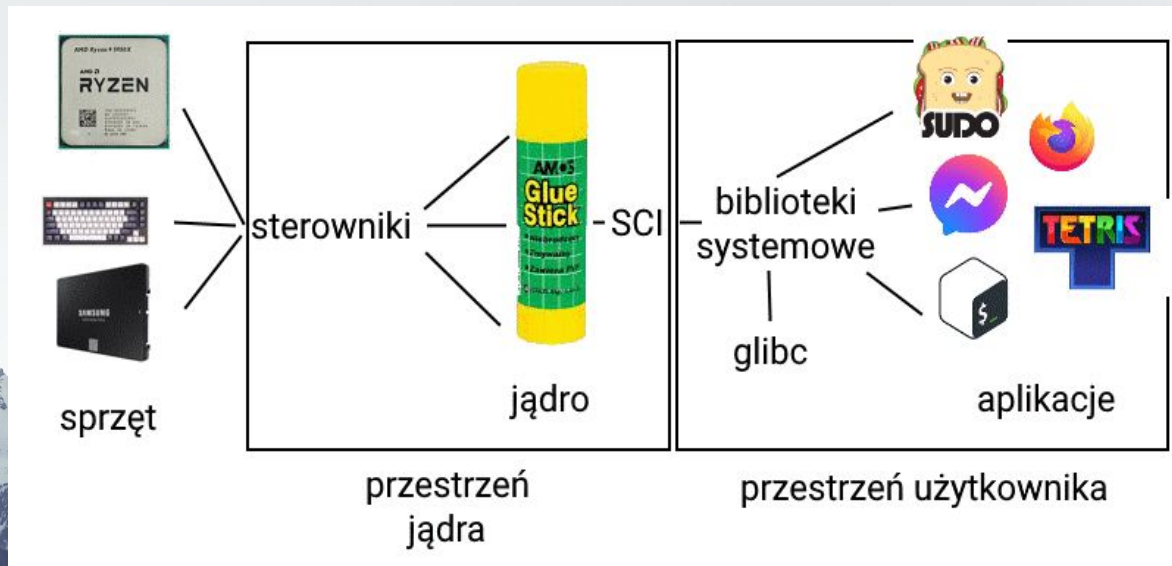


# Co to właściwie jest ten Linux?

To nie tylko żargon dla hipsterów, ale również określenie na rodzinę systemów operacyjnych posiadających to samo jądro, czyli właśnie Linux.

Jądro pełni najważniejszą rolę, czyli zespolenie sprzętu z procesami systemowymi, co umożliwia działanie aplikacji.

Mówiąc o Linuksie, najczęściej myślimy o projekcie tzw. GNU + Linux, czyli jądrem, biblioteką C i podstawowymi narzędziami z dawnego UNIX-a.



[illegible][illegible]

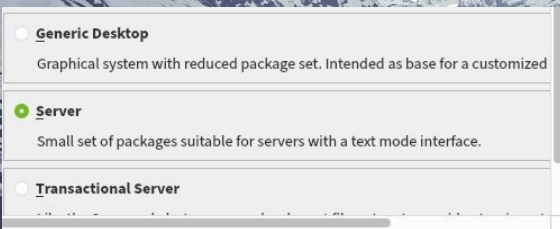
zaimplementować wszystkie możliwe funkcje i niepotrzebne śmieci.

```
Hackerman  
Hackerman  
Hackerman  
1494 root    20  0  300M 50304 40320 S 0  
1867 root    20  0  580M 48256 42496 S 0  
1890 root    20  0  580M 48256 42496 S 0  
1891 root    20  0  580M 48256 42496 S 0  
Hackerman  
2069075 root   20  0  480M 29440  8320 S 0  
2069077 root   20  0  480M 29440  8320 S 0  
Hackerman  
2069080 root   20  0  480M 29440  8320 S 0  
Hackerman  
2069286 root   20  0  480M 29440  8320 S 0  
1424 polkitd  20  0  1863M 28096 16832 S 0  
Hackerman  
1481 polkitd  20  0  1863M 28096 16832 S 0  
1483 polkitd  20  0  1863M 28096 16832 S 0  
Hackerman  
1486 polkitd  20  0  1863M 28096 16832 S 0
```

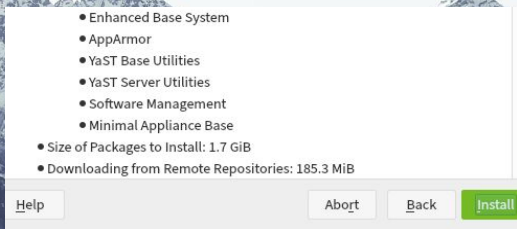
F1Help F2Setup F3SearchF4FilterF5Tree F6SortBy



Właściwie wszystkie **dystrybucje** instaluje się w podobny sposób. Po sformatowaniu dysku, wybraniu języka, czasu i lokalizacji system kopiuje wszystko do pamięci a na koniec instaluje **bootloader**, czyli program, który ładuje inne potrzebne elementy. Przykładowo zainstalujemy **openSUSE**, opartego na Slackware - pierwszej poważnej dystrybucji Linuksa. Plik instalacyjny pobieramy ze strony <https://get.opensuse.org/tumbleweed>. Do wszystkiego wykorzystamy VirtualBox'a.

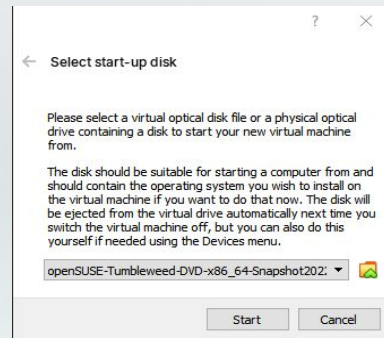
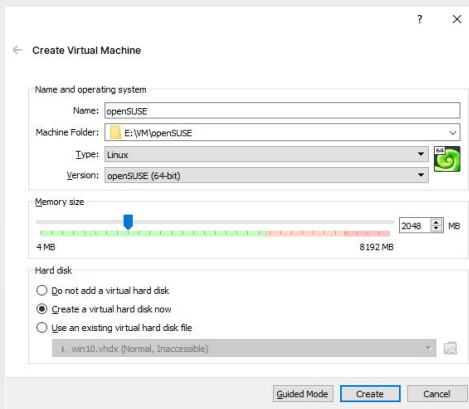


Większość opcji w menu możemy bezpośrednio pominąć. Oczywiście istotne jest wybranie odpowiedniego pakietu oprogramowania. Korzystamy z maszyny wirtualnej, więc nie musimy się martwić o partycjonowaniu dysku.



# Instalacja

Tworzymy maszynę, a potem dysk, na przykład o dynamicznej wielkości 20GB. Na koniec ładujemy wcześniej pobrany plik .ISO.

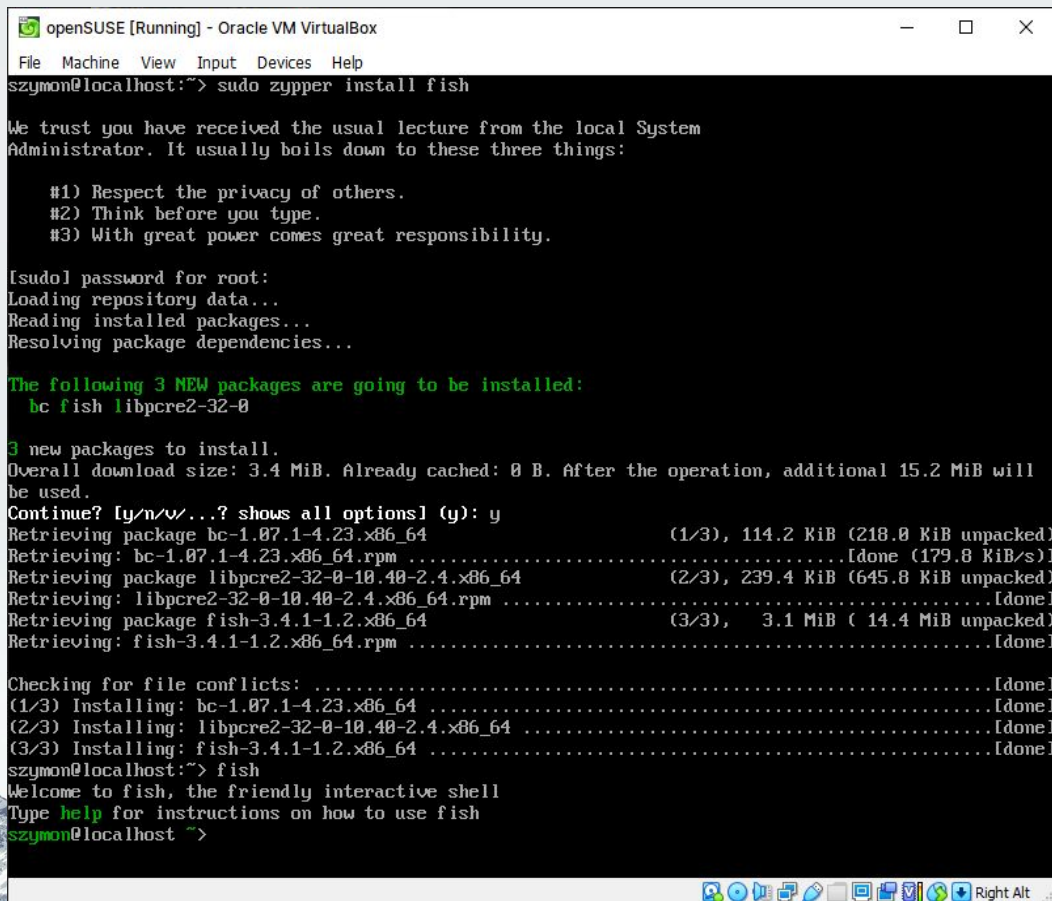


Ważnym krokiem w instalacji jest również kliknięcie przycisku rozpoczynającego instalację.

# A teraz co?

Jeżeli kliknięcie przycisku 'zainstaluj' poszło pomyślnie, możemy zrestartować maszynę wirtualną i przejść do powłoki, czyli zwykłego wiersza poleceń. Istnieją dystrybucje typowo serwerowe, które dają nam więcej możliwości administracji graficznej, bynajmniej nasz **openSUSE**, ale stosując takie podejście nie nauczymy się zbyt wiele o Linux'ie. Rozwiązywanie problemów związanych z programami albo sterownikami zazwyczaj sprowadza się do ręcznego edytowania rzeczy w terminalu!

Każda dystrybucja daje nam inny sposób na instalowanie programów. W naszym przypadku jest to **zypper**. Przykładowo możemy ściągnąć alternatywną powłokę **fish** - jest kolorowa i podpowiada nam wiele rzeczy po drodze.



```
openSUSE [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
szymon@localhost:~> sudo zypper install fish

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for root:
Loading repository data...
Reading installed packages...
Resolving package dependencies...

The following 3 NEW packages are going to be installed:
  bc fish libpcre2-32-0

3 new packages to install.
Overall download size: 3.4 MiB. Already cached: 0 B. After the operation, additional 15.2 MiB will
be used.
Continue? [y/n/w/...? shows all options] (y): y
Retrieving package bc-1.07.1-4.23.x86_64 ..... (1/3), 114.2 KiB (218.0 KiB unpacked)
Retrieving: bc-1.07.1-4.23.x86_64.rpm .....[done (179.8 KiB/s)]
Retrieving package libpcre2-32-0-10.40-2.4.x86_64 ..... (2/3), 239.4 KiB (645.8 KiB unpacked)
Retrieving: libpcre2-32-0-10.40-2.4.x86_64.rpm .....[done]
Retrieving package fish-3.4.1-1.2.x86_64 ..... (3/3), 3.1 MiB (14.4 MiB unpacked)
Retrieving: fish-3.4.1-1.2.x86_64.rpm .....[done]

Checking for file conflicts: .....[done]
(1/3) Installing: bc-1.07.1-4.23.x86_64 .....[done]
(2/3) Installing: libpcre2-32-0-10.40-2.4.x86_64 .....[done]
(3/3) Installing: fish-3.4.1-1.2.x86_64 .....[done]
szymon@localhost:~> fish
Welcome to fish, the friendly interactive shell
Type help for instructions on how to use fish
szymon@localhost ~>
```

# A teraz co? - część 2.

Warto pamiętać, że **fish** nie jest prawdziwym zamiennikiem dla domyślnej powłoki (**bash**), bo nie jest do końca kompatybilny z obecnymi standardami. Jednak znacznie ułatwia pracę przy Linuksie i nie musimy go nadmiernie konfigurować, aby z niego korzystać. Nie zależnie od tego, co mamy, warto znać podstawowe komendy i własności powłoki.

```
szymon@localhost ~$ echo "ipsum" > ipsum
szymon@localhost ~$ echo "lorem" > lorem
szymon@localhost ~$ ls
bin/ ipsum lorem
szymon@localhost ~$ mkdir lorem-ipsum
szymon@localhost ~$ cd lorem-ipsum/
szymon@localhost ~/lorem-ipsum$ mv ../{ipsum,lorem} .
szymon@localhost ~/lorem-ipsum$ ls
ipsum lorem
szymon@localhost ~/lorem-ipsum$ cat ipsum; cat lorem
ipsum
lorem
szymon@localhost ~/lorem-ipsum$ cat lorem >> lorem_ipsum && cat ipsum >> lorem_ipsum
szymon@localhost ~/lorem-ipsum$ cat lorem_ipsum
lorem
ipsum
szymon@localhost ~/lorem-ipsum$ cat lorem_ipsum | grep ipsum
ipsum
szymon@localhost ~/lorem-ipsum$ find /home/szymon/ | grep lorem
/home/szymon/lorem-ipsum
/home/szymon/lorem-ipsum/ipsum
/home/szymon/lorem-ipsum/lorem
/home/szymon/lorem-ipsum/lorem_ipsum
szymon@localhost ~/lorem-ipsum$
```

Właściwie wszystko w Linuksie jak i innych systemach podobnych do UNIX'a jest traktowane jako plik: skrypty, konfiguracje systemowe, a nawet pliki! Przebywając w wierszu poleceń właściwie wszystko jest traktowane jako **strumień** ( stdio i stdout, podobne do słynnego **iostream** ). Kiedy przesyłamy aplikacji jakiś plik, tak naprawdę przesyłamy jego zawartość jako ciąg tekstu. Każdy program przyjmuje argumenty: np. **rm -r**, **ls -la**. Opcja **-h** albo **--help** wyświetla komunikat pomocny - daje nam informacje o autorze i jak korzystać z aplikacji. Niestety to rozwiązanie zakłada umiejętność czytania ze zrozumieniem po angielsku.

Ściąga hieroglifów Linuksowych:

**cd** – zmiana folderu  
**mv** – przemieszczanie pliku lub folderu w inne miejsce  
**cp** – kopiowanie pliku gdzieś indziej  
**rm** – usuwanie pliku  
**ls** – wyświetlenie zawartości folderu  
**sudo** – uruchamianie programu jako administrator, jeśli znamy do niego hasło  
**mkdir** – tworzenie nowego folderu  
**sync** – synchronizowanie zawartości folderów ( **cp -r** na sterydach )  
**grep** – wyszukiwanie słów  
**find** – rekursywne znajdowanie plików w danym folderze

" ." - obecny folder        ./  
" .. " - folder nadrzędny .././home  
" / " - magiczny dzielnik między folderami  
" | " - przekazanie strumienia do innego programu        **find ./ | grep lorem**  
" > " - nadpisanie pliku  
" >> " - dodać do pliku  
" && " - wykonanie kolejnej czynności, tylko wtedy, gdy poprzednia pomyślnie się zakończyła  
" ; " - wykonanie kolejnej czynności na pałę  
" \* " - wszystkie rzeczy zawarte w danej rzeczy ( rozwiązanie mało praktyczne )



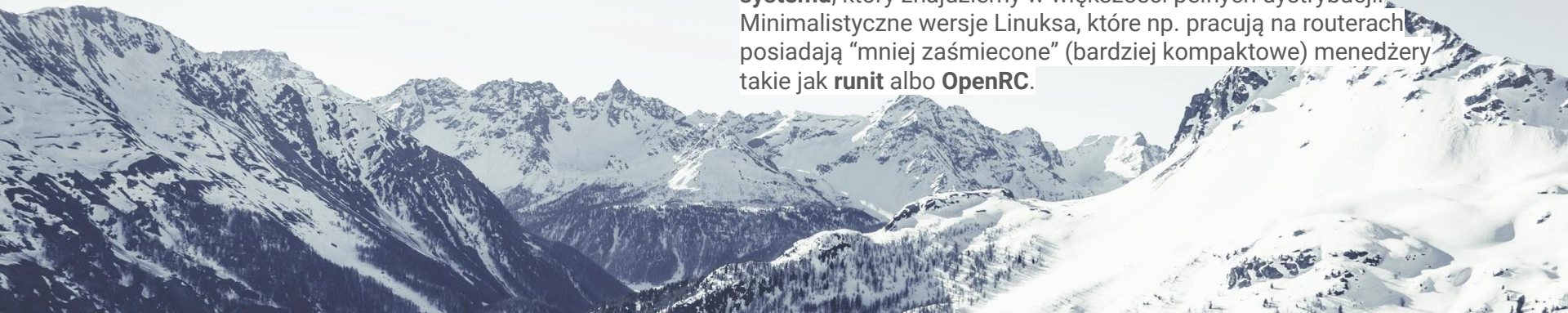
# Podstawy podstaw: serwery

Spora część dzisiejsze infrastruktury w sieci opiera się na programie **NGINX**. Tak jak wiele rzeczy w IT jest on wspierany przez firmę i rozwijany jako projekt z otwartym źródłem. Ma on poważne zastosowania takie jak: serwer plików, serwer proxy, system zarządzania obciążeniem, szyfrowanie połączeń i tym podobne. Oczywiście to wszystko jest przydatne, ale więcej można się nauczyć tworząc coś prostszego - na przykład serwer do gry Minecraft. NGINX może jednak przydać się jeżeli chcielibyśmy podpiąć do niego jakąś domenę.

```
szymon@localhost ~$ chsh -s /usr/bin/fish; fish
Password:
Welcome to fish, the friendly interactive shell
Type help for instructions on how to use fish
szymon@localhost ~$ sudo passwd root
New password:
Retype new password:
passwd: password updated successfully
szymon@localhost ~$ _
```

Najpierw możemy użyć przydatnej komendy: "**chsh -s /usr/bin/fish**" po to, aby zmienić domyślną powłokę z **bash** na **fish**. Potem ustawiamy hasło dla najważniejszego konta (**root**). Przez to możemy wywołać komendę **sudo** - która sama w sobie wykonuje inne komendy jako administrator. Teraz możemy wpisać magiczną komendę "**zypper in nginx**". Po zaakceptowaniu warunków nieświadczenia usług program pokazuje postęp w instalacji.

Zanim pójdziemy zdawać certyfikaty na inżyniera sieciowego w CISCO, warto umieć ustawić podstawowy serwer. **systemctl** jest narzędziem, które służy do zarządzania programów pracujących w tle. Bardziej specyficznie jest to interfejs menedżera systemu **systemd**, który znajdziemy w większości pełnych dystrybucji. Minimalistyczne wersje Linuksa, które np. pracują na routerach posiadają "mniej zaśmiecone" (bardziej kompaktowe) menedżery takie jak **runit** albo **OpenRC**.



# Serwery cz. 2 - plan i teoria

W następnym kroku zainstalujemy środowisko **Javy** ( **C++** na drożdżach ) i serwer **Minecraft'a**. Jeżeli posiadalibyśmy domenę, moglibyśmy ją podpiąć pod serwer proxy ( np. NGINX ). Większość z tego zrobimy ręcznie, bo jak się okazuje, "gotowe" rozwiązania nie zawsze są adekwatne do naszych potrzeb, a często bezużyteczne i frustrujące ( czyt. Docker albo Snap ). Przy okazji nauczymy się obsługi archiwów i instalacji usług. W naszej architekturze sieci możemy zamienić Minecraft'a na właściwie jakiegokolwiek program który obsługuje protokoły TCP albo UDP (bardziej wysublimowaną wersję poprzednika), a niezmienna konfiguracja NGINX'a umożliwi nam dodanie domeny i przekierowań.

Wszystkie systemy oparte na UNIX'ie mają podobną hierarchię plików i folderów. `" / "` jest punktem podstawowym dla systemu. `" /home "` zawiera folder dla każdego użytkownika w systemie, `" /var "` dzienniki dla programów (np. błędy przez nie napotkane), `" /bin /lib /sbin "` ważne programy i biblioteki systemowe, `" /usr/(share, lib, local) "` głównie biblioteki i bardziej skomplikowane programy.

Tradycyjnie projekty stworzone przez użytkownika, takie jak skrypty do serwerów umieszcza się w folderze `" /srv "`, chociaż nie ma to znaczenia z punktu widzenia systemu.





# Konfiguracja sieci

Najistotniejszym krokiem jest skonfigurowanie sieci. VirtualBox domyślnie ustawia maszyny w sieci NAT, analogicznej do routera - tylko niektóre porty są aktywne. Najpierw przestawiamy sieć z NAT na adapter mostkowy (bridge), który umożliwia pełną kontrolę nad przepływem danych.

Równie ważne jest ustawienie zapory sieciowej (tak zwanego firewall'a). W przypadku serwerów ala RedHat albo naszym openSUSE jest to **firewall-cmd**. Zapora pełni prostą funkcję - zezwala na łączenie się z tylko niektórymi portami, czyli magicznymi liczbami, które same w sobie nic nie znaczą. W przypadku Minecraft'a musimy wpisać nic nieznaczącą liczbę **25565**. Argumenty **--permanent** i **--zone=public** są uniwersalne do naszych celów.

```
szymon@pc-227 /e/n/vhosts.d> sudo firewall-cmd --zone=public --permanent --add-port=25565/tcp
success
```

Musimy też wiedzieć, jaki adres IP użyć w celu połączenia się z naszym serwerem. Komenda "ip" pokazuje 2 urządzenia: nr 1 odnosi się do wirtualnego adaptera o adresie **127.0.0.1**, do którego ma dostęp wyłącznie komputer. W naszym przypadku Linux jest podłączony do tylko jednej karty sieciowej emulowanej przez VirtualBox'a - ma ona adres **192.168.1.\*\*\***. Za pomocą tego IP będziemy mogli połączyć się z serwerem z sieci prywatnej komputera, czyli wszystkich komputerów w obrębie jednego routera.

Urządzenia Pomoc

Napędy optyczne  
Audio  
Sieć  
USB  
Kamerki internetowe  
Udzielanie folderu

Ustawienia sieciowe...

Podłącz kartę sieciową

Network

Karta 1 Karta 2 Karta 3 Karta 4

☒ Włącz kartę sieciową

Podłączona do: NAT

Nazwa:

Zaawansowane

Network

Karta 1 Karta 2 Karta 3 Karta 4

☒ Włącz kartę sieciową

Podłączona do: Mostkowana karta sieciowa (bridg

Nazwa: Intel(R) Ethernet Connection I217-V

Zaawansowane

```
szymon@pc-227 /root> ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:b5:47:4e brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.26/24 brd 192.168.1.255 scope global dynamic noprefixroute enp0s3
        valid_lft 84525sec preferred_lft 84525sec
    inet6 fe80::a00:27ff:feb5:474e/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

# Ustawianie wszystkiego

Przed dalszym konfigurowaniem czegokolwiek musimy zainstalować środowisko Javy, np. wersję 17 z długoterminowym wsparciem.

```
szymon@pc-227 /s/minecraft> sudo zypper in java-17-openjdk_
```

W końcu możemy zacząć serwer - najpierw tworząc folder, a potem przywłaszczając go danemu użytkownikowi za pomocą komendy "**chown -R** użytkownik ."

```
szymon@pc-227 /srv> sudo mkdir minecraft
szymon@pc-227 /srv> cd minecraft/ && sudo chown -R szymon .
```

W tej prezentacji będziemy instalować wersję z Forge, która umożliwia dodawanie modyfikacji do gry. Po zlokalizowaniu dowolnej wersji możemy pobrać archiwum **.jar** (wersja pliku wykonywalnego w Jav'ie). Do tego wykorzystamy komendę "**wget link**"

Następnie wpisujemy magiczną komendę: "**java -jar plik --installServer**".

Teraz posiadamy pełny folder i gotowy skrypt do uruchomienia (**run.sh**). Nie potrzebujemy już instalatora więc możemy ten plik usunąć.

```
szymon@pc-227 /s/minecraft> ls -la
total 7496
drwxr-xr-x 1 szymon root      218 Jun  8 10:49 ./
dr-xr-xr-x 1 root  root      40 Jun  8 10:46 ../
-rw-r--r-- 1 szymon szymon 6661580 Jun  8 10:47 forge-1.18.2-40.1.0-installer.jar
-rw-r--r-- 1 szymon szymon 995999 Jun  8 10:50 forge-1.18.2-40.1.0-installer.jar.log
drwxr-xr-x 1 szymon szymon   168 Jun  8 10:49 libraries/
-rw-r--r-- 1 szymon szymon   362 Jun  8 10:49 run.bat
-rwxr--r-- 1 szymon szymon   365 Jun  8 10:49 run.sh*
-rw-r--r-- 1 szymon szymon   339 Jun  8 10:49 user_jum_args.txt
szymon@pc-227 /s/minecraft> rm forge-1.18.2-40.1.0-installer.jar
szymon@pc-227 /s/minecraft>
```

Komendy uruchamiające serwer są zawarte w skrypcie bash'a **run.sh**. Po pierwszym jego uruchomieniu utworzy kilka ważnych folderów i plików, ale za nim przejdziemy dalej musimy zedytować plik **eula.txt** - tak, nawet w świecie serwerów musimy zgadzać się na warunki usług :D. Program możemy zatrzymać za pomocą skrótu **Ctrl+C**. Do samego edytowania wykorzystamy standardowego edytora tekstowego: "**nano eula.txt**". Po zmianie **false** na **true** możemy wyjść z edytora wciskając **Ctrl+X -> y -> Enter**.

```
GNU nano 6.3 eula.txt
#By changing the setting below to TRUE you are indicating your agreement to o
#Wed Jun 08 11:15:55 EDT 2022
eula=false
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify
```

# Ustawianie wszystkiego cz. 2

To, na co się przed chwilą zgodziliśmy bez myślenia to **EULA**, czyli warunki twórców gry m.in. zabraniające sprzedaży zawartych w niej przedmiotów za prawdziwe pieniądze (w końcu oni muszą wyłącznie czerpać zyski) (w końcu to Microsoft :D).

Z racji tego, że zainstalowaliśmy Forge'a, możemy dodać do gry modyfikacje, których można znaleźć tysiące w Internecie, np. Formie pliku **zip** (ściągamy ponownie używając **wget**), który możemy rozpakować używając **unzip**.

```
szymon@pc-227 /s/m/mods> wget https://media.minecraftforge.net/files/3
szymon@pc-227 /s/m/mods> unzip Create+Flavored+3.3+SERVER.zip
```

Zanim ponownie uruchomimy nasz serwer, możemy zmienić ustawienia w pliku konfiguracyjnym **server.properties** - jest on pełen żargonu Minecraft'owego - seed, PvP, gamemode i tym podobne. Kiedy znowu włączymy **run.sh**, z pewnością najwięcej czasu zajmie generacja świata, ale w końcu będziemy mieć dostęp do konsoli serwera.

Na koniec możemy napisać skrypt, który automatycznie włącza nasz serwer w tle za pomocą programu **tmux**.

```
szymon@pc-227 /s/minecraft> sudo zypper in tmux
```

Wpisując "**tmux**" otrzymujemy nowe "okno" w konsoli i stąd uruchamiamy **run.sh** tak jak wcześniej.

```
Welcome to fish, the friendly interactive shell
Type help for instructions on how to use fish
szymon@pc-227 /s/minecraft> ./run.sh
```

```
[0] 0:fish* "[pc-227] /s/minecraft" 07:59 09-Jun-22
```

Możemy stąd wyjść wciskając **Ctrl + B** i następnie **D**, a wejść z powrotem komendą "**tmux attach**".



# Co dalej? - Pisanie oprogramowania

Jeżeli do tej pory wszystko płynnie się udało, posiadamy teraz w pełni sprawny serwer Minecraft'a. Wystarczy uruchomić maszynę, otworzyć nową sesję w konsoli i wpisać `./run.sh`. Moglibyśmy również zainstalować środowisko graficzne, chociaż w przypadku serwerów nie ułatwia to nam szczególnie pracy.

Najciekawszą możliwością, którą daje nam Linux jest tworzenie własnych aplikacji opartych o systemowe biblioteki. Jednym z założonych projektów stypendialnych jest właśnie taki program - narzędzie do redukcji szumów w zdjęciach. Tego rodzaju projekty najczęściej przechowujemy w chmurze - w portalach takich jak **GitHub**. Aby zarządzać, a w naszym przypadku skopiować repozytorium (projekt) wykorzystamy narzędzia **git**.

```
szymon@pc-227 ~> cd /home/szymon/
szymon@pc-227 ~> sudo zypper in git
szymon@pc-227 ~> git clone https://github.com/Iquerno/szumovski
Cloning into 'szumovski'...
```

Tego rodzaju projekty z reguły zawsze potrzebują jakiejś biblioteki, czyli zbioru innych programów, do poprawnego działania. W tym przypadku jest to **Magick++**, która służy do manipulacji obrazów.

```
szymon@pc-227 ~/szumovski (master)> sudo zypper in libMagick++-devel
```

Ostatnim krokiem jest skompilowanie samego programu. Najczęściej spotkamy się z plikiem **Makefile**, który zawiera gotowe skrypty na zainstalowanie programu. Pliki wykonywalne w Linux'ie nie mają rozszerzenia, więc gotowy program możemy uruchomić wpisując `./bin/szumovski`

```
szymon@pc-227 ~/szumovski (master)> ls -l
total 8
drwxr-xr-x 1 szymon szymon 18 Jun 13 09:57 bin/
drwxr-xr-x 1 szymon szymon 26 Jun 13 08:24 library/
-rw-r--r-- 1 szymon szymon 296 Jun 13 09:56 Makefile
-rw-r--r-- 1 szymon szymon 93 Jun 13 08:24 README.txt
drwxr-xr-x 1 szymon szymon 16 Jun 13 08:24 source/
szymon@pc-227 ~/szumovski (master)> _
```

```
szymon@pc-227 ~/szumovski (master)> sudo cp bin/szumovski /usr/local/bin
```

Teraz kopiujemy plik do folderu `/usr/local/bin` zawierającej inne aplikacje. W ten sposób możemy uruchomić **szumovski** z jakiegokolwiek ścieżki. || Każda aplikacja przyjmuje argumenty (opcje) w postaci np. `"-i ściezka --sila liczba"`. Dodatkowo możemy je sprawdzić wpisując `"-h"` albo `"--help"`. W tej chwili wystarczy wpisać komendę i *voilà!*

```
szymon@pc-227 ~> szumovski -i for_sale.png -s 6
```

Jak widać parędziesiąt linijek **C++** czasem dobrze popłaca.



Fotografowie go nienawidzą

Jeden prosty trik i ... zero szumów!

**SZUMOVSKI**

**Dziękuję za uwagę!**

# Bibliografia

1. [pl.wikipedia.org](https://pl.wikipedia.org)
2. metoda prób i błędów
3. wiedza
4. [opensuse.org](https://opensuse.org)

