

# Wstęp do systemu Linux / Mac

v.1.0

# Plan

- Wprowadzenie do systemów Unix
  - Poruszanie się po konsoli, system plików
  - Użytkownicy
  - Dostęp do plików
- Instalowanie nowych programów
  - Tematy zaawansowane
  - Różnorodność w systemach Linux

# Uruchamianie terminala

# Terminal

## Czym jest terminal?

**Terminal** (tak właściwie jest to emulator terminala) jest interfejsem, dzięki któremu możemy wpisywać i wykonywać komendy tekstowe.

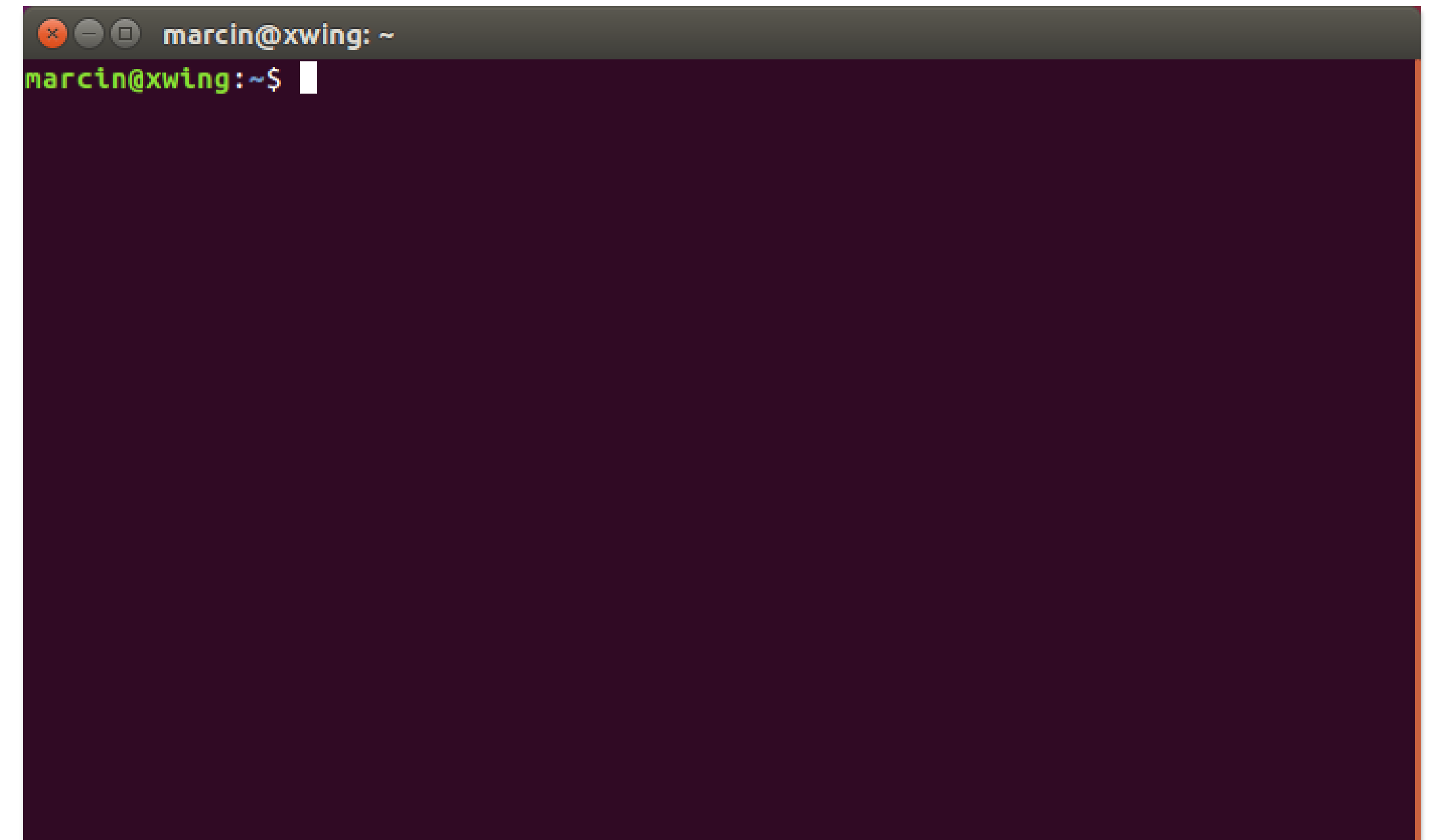
Praca w terminalu pozwala nam wykonywać zadania dużo szybciej niż przy użyciu interfejsów graficznych. Dodatkowo w większości przypadków mamy dużo większe pole manewru oraz więcej opcji przy wykonywaniu zadań.

**Znajomość systemów operacyjnych oraz obsługi terminala jest umiejętnością, którą powinien posiadać każdy programista!**

# Jak otworzyć terminal?

Jeśli jesteś użytkownikiem Ubuntu:

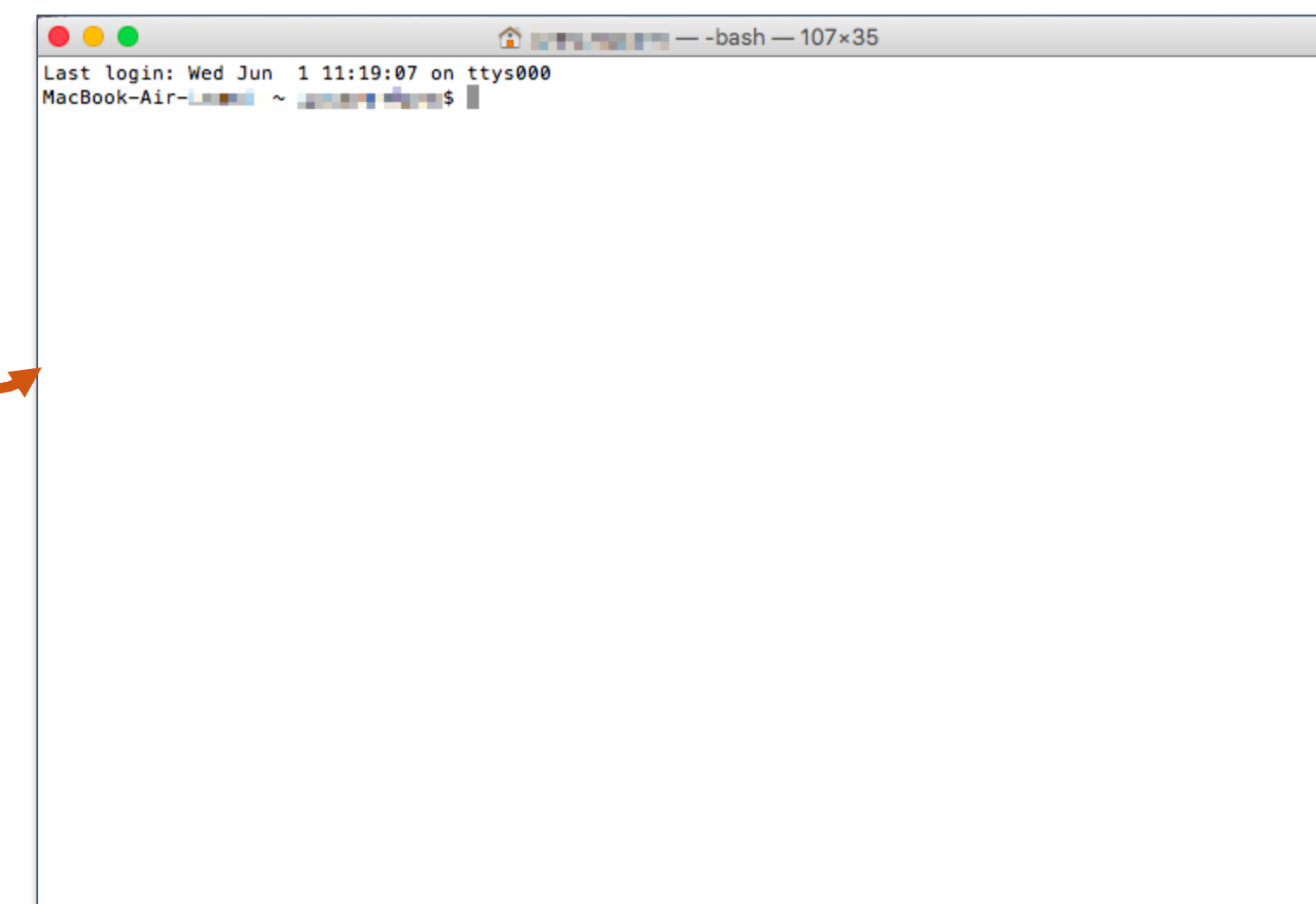
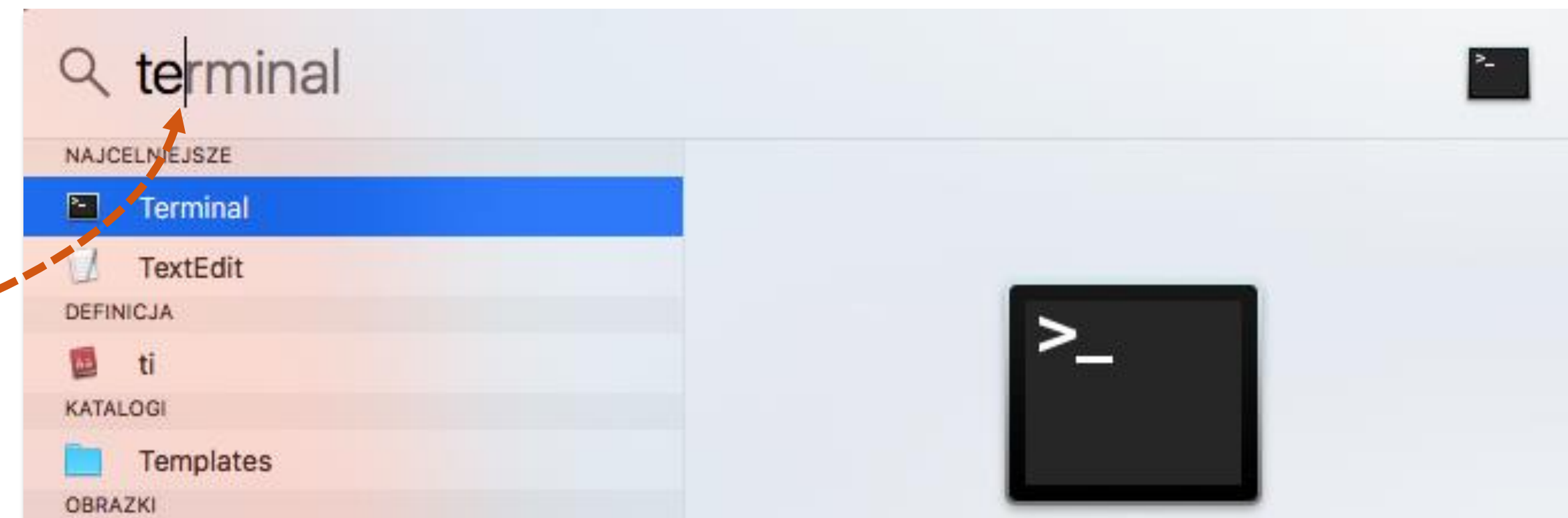
Otwórz terminal kombinacją klawiszy  
**Ctrl-Alt-T**



# Jak otworzyć terminal?

Jeśli jesteś użytkownikiem Mac OS:

- Wciśnij kombinację klawiszy **Cmd+Spacja**,
- W oknie **Spotlight** zacznij wpisywać „**terminal**”,
- Potwierdź klawiszem **Enter**.



# Wprowadzenie do systemów Unix

# Wprowadzenie do systemów Unix

## Krótko o Linuksie

- Linux to rodzina uniksopodobnych systemów operacyjnych.
- Linux jest jednym z przykładów wolnego i otwartego oprogramowania.
- Jego kod źródłowy może być dowolnie wykorzystywany, modyfikowany i rozpowszechniany.
- Systemy operacyjne Maców są oparte na podobnej rodzinie – stąd większość komend będzie działać dokładnie tak samo.

## Co to jest shell?

- Shell jest najniższą powłoką interfejsu użytkownika typową dla systemów uniksowych.
- Jest to część systemu odpowiedzialna za podstawową interakcję z użytkownikiem.
- Każdy z shelli musi implementować podstawową liczbę komend wymaganych przez standard.
- Najczęściej każdy z nich usprawnia standard przez rozszerzenie liczby poleceń.



# Przydatne skróty

CTRL + C	Przerywanie pracy procesu
CTRL + D	Wysyła sygnał EOF (end-of-file)
CTRL + R	Przeszukuje historię pod względem podanych liter
CTRL + Z	Zatrzymanie procesu
CTRL + A	Przeniesienie kursora na początek linii

# Wprowadzenie do systemów Unix

## Historia poleceń

Powłoka shell zapamiętuje ostatnio używane komendy (domyślnie – 1000):

- **history** – pokazuje listę używanych komend,
- **!!** – wykonuje ostatnią komendę,
- **!-3** – wykonuje trzecią komendę od końca z listy,
- **!5** - wykonuje piątą komendę z listy,
- **!grep** – wywołuje ostatnią komendę zaczynającą się od **grep**,

W nowszych shellach do wywołania historii poleceń służy skrót klawiszowy: **CTRL + R**.

## Edytory tekstu

- **Gedit** – podstawowy (zainstalowany od początku) edytor tekstu w Ubuntu. Ma wspomaganie dla systemu kodowania UTF-8.
- **Vi** – podstawowy edytor tekstu w konsoli. Nieporęczny i trudny ale użyteczny. Warto jednak nauczyć się jego obsługi, jeżeli pracujemy przez SSH. Niezastąpiony przy pracy z wielkimi plikami.
- **Geany** – słynny multiplatformowy edytor tekstu. Bogata liczba opcji czyni go jednym z lepszych edytorów dla programistów.

# Edytor Vi

## Podstawy użytkowania Vi

**Vi** działa w dwóch trybach:

- komend – tryb, w którym wpisujemy komendy programu (takie jak: zapisz plik, zamknij program itp.). Żeby z niego przejść do trybu edycji należy wcisnąć klawisz **I** (od słowa „insert”).
- edycji – tryb, w którym mamy możliwość edycji pliku. Żeby z niego przejść do trybu komend wciskamy klawisz **ESC**.

## Podstawowe komendy Vi

<b>:30</b>	Przesuwa kursor do wskazanej linii
<b>/&lt;ciąg_znaków&gt;</b>	Wyszukuje dany napis, np. /anything
<b>? &lt;ciąg_znaków&gt;</b>	Wyszukuje dany napis wstecz (od końca pliku)
<b>n</b>	Znajduje następne wystąpienie danego wyszukiwania
<b>N</b>	Znajduje poprzednie wystąpienie danego wyszukiwania
<b>:e &lt;nazwa_pliku&gt;</b>	Otwiera nowy plik o podanej nazwie
<b>:w</b>	Zapisuje plik
<b>:w!</b>	Zapisuje plik, nadpisując pozwolenia tego pliku (zdejmuje read-only)
<b>:w &lt;nazwa_pliku&gt;</b>	Zapisuje do nowego pliku o podanej nazwie
<b>:q</b>	Wychodzi z programu

**Poruszanie się  
po konsoli,  
system plików**

# Struktura katalogów w systemie Linux

<b>/</b>	Główny katalog w systemie (wszystkie katalogi są podkatalogami / )
<b>/dev</b>	Katalog, w którym znajdują się wszystkie urządzenia
<b>/proc</b>	Katalog wymiany danych komunikacji międzyprocesowej, zawiera też szczególne informacje na temat systemu. Nie zawiera w sobie żadnego „realnego” pliku
<b>/etc</b>	Katalog zawierający pliki konfiguracyjne
<b>/sbin</b>	Katalog zawierający podstawowe pliki binarne potrzebne do działania systemu
<b>/lib</b>	Katalog zawierający biblioteki zainstalowane w systemie
<b>/mnt</b>	Katalog, w którym montowane są wszystkie dyski
<b>/bin</b>	Katalog zawierający programy

<b>/etc</b>	Pliki używane przez podsystemy Uniksa (np. bazy danych)
<b>/etc/init.d</b>	Katalog zawierający skrypty uruchamiane podczas startu systemu
<b>/etc/profile.d</b>	Katalog zawierający skrypty uruchamiane przy logowaniu danego użytkownika ( )
<b>/home</b>	Katalog domowy użytkownika
<b>/root</b>	Katalog domowy użytkownika root (głównego użytkownika systemu)
<b>/tmp</b>	Katalog zawierający pliki chwilowe potrzebne do działania programów i systemu
<b>/usr</b>	Katalog zawierający pliki wykonywalne programów, kod źródłowy, biblioteki i dokumentacje



# Struktura katalogów w systemie Mac

<b>/</b>	Główny katalog w systemie (wszystkie katalogi są podkatalogami / )
<b>/Applications</b>	Katalog, w którym instalowane są aplikacje
<b>/Volumes</b>	Katalog, w którym montowane są wszystkie dyski (w tym pliki dmg z aplikacjami) np. cdrom
<b>/etc</b>	Katalog zawierający pliki konfiguracyjne
<b>/sbin</b>	Katalog zawierający podstawowe pliki binarne potrzebne do działania systemu
<b>/Library</b>	Katalog zawierający biblioteki zainstalowane w systemie
<b>/bin</b>	Katalog zawierający programy

<b>/etc</b>	Pliki używane przez podsystemy Maca (np. bazy danych)
<b>/dev</b>	Katalog, w którym znajdują się wszystkie urządzenia
<b>/etc/profile</b>	Katalog zawierający skrypty uruchamiane przy logowaniu danego użytkownika
<b>/Users</b>	Katalogi domowe użytkowników
<b>/System</b>	Katalog zawierający pliki systemowe
<b>/tmp</b>	Katalog zawierający pliki chwilowe potrzebne do działania programów i systemu
<b>/usr</b>	Katalog zawierający pliki wykonywalne programów, kod źródłowy, biblioteki i dokumentacje

# Podstawowe komendy – pliki i katalogi

<b>ls</b>	Wyświetla wszystkie pliki	<b>ls -a</b> <b>ls -l</b>	Wyświetla także pliki ukryte Wyświetla dodatkowe informacje
<b>mkdir</b> dirname	Tworzy katalog		
<b>cd</b> dirname	Przechodzi do wskazanego katalogu	<b>cd .</b> <b>cd ..</b> <b>cd ~</b>	Obecny katalog Katalog bezpośrednio wyżej Katalog domowy
<b>pwd</b>	Wyświetla ścieżkę do katalogu, w którym się znajdujemy		
<b>cp</b> file1 file2	Kopiuje file1 na miejsce file2		
<b>mv</b> file1 file2	Przenosi file1 na miejsce file2		

# Podstawowe komendy – pliki i katalogi

<b>rm</b> file	Usuwa plik	<b>rm -r</b>	Usuwa także katalogi
<b>rmdir</b> dirname	Usuwa katalog		
<b>cat</b> file	Wyświetla wskazany plik		
<b>less</b> file	Wyświetla plik strona po stronie		
<b>head</b> file	Wyświetla pierwsze 10 linii pliku	<b>head -n</b>	Wyświetla <b>n</b> linii
<b>tail</b> file	Wyświetla ostatnie 10 linii pliku	<b>tail -n</b>	Wyświetla <b>n</b> linii
<b>wc</b> file	Podaje liczbę słów, znaków, linii lub bajtów w pliku (lub potoku)	<b>wc -c</b> <b>wc -l</b> <b>wc -w</b> <b>wc -m</b>	liczba bitów, liczba bajtów, liczba słów, liczba znaków
<b>touch</b> file	Tworzy plik o podanej nazwie		



# Podręczniki systemowe

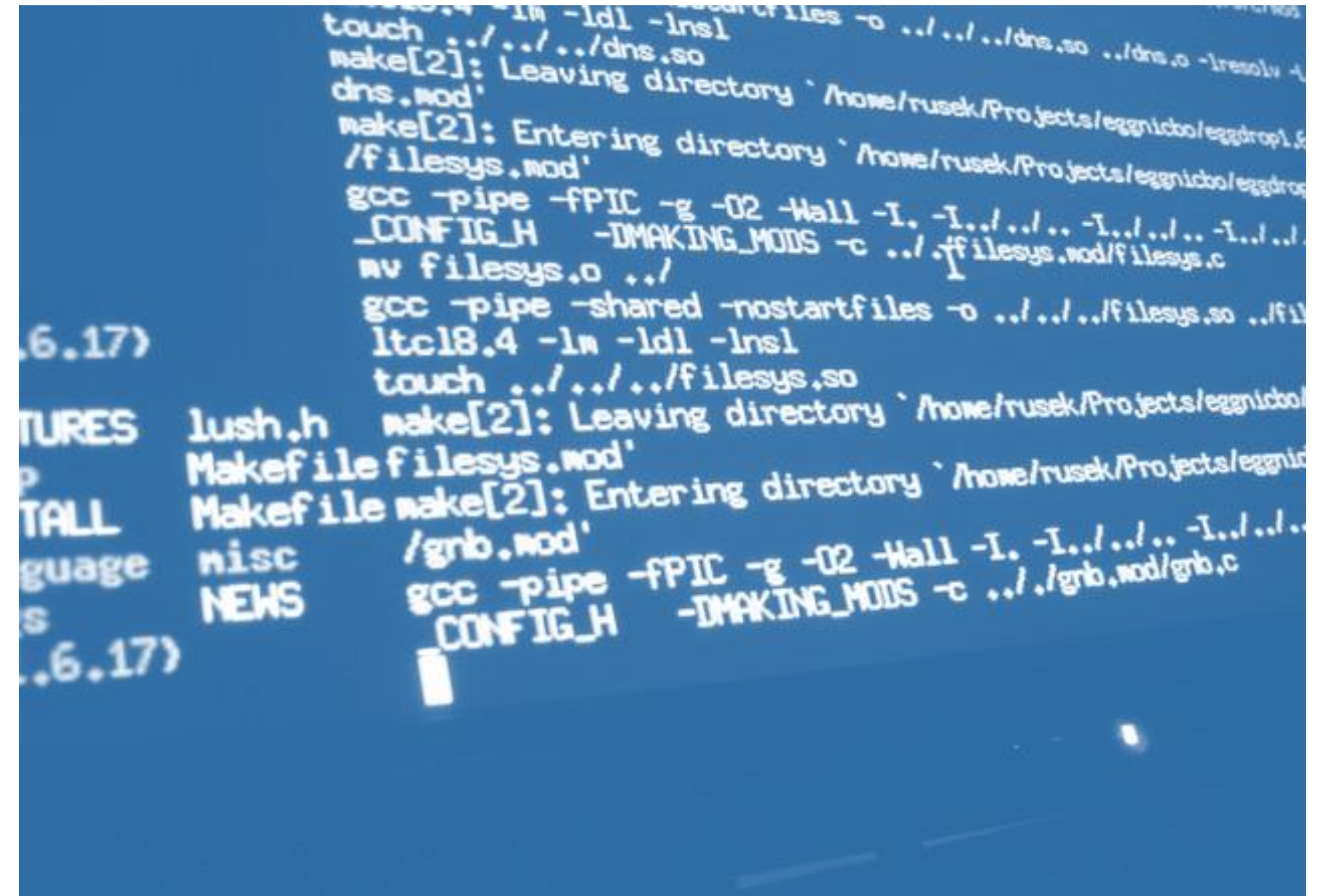
**man** <nazwa\_komendy> – otwiera podręcznik pomocy danej komendy np. **man** ls.

Komenda z jednym z tych parametrów wyświetla dodatkowe instrukcje:

<nazwa\_komendy> **--help**

<nazwa\_komendy> **-h**

**apropos** com – wyświetla wszystkie komendy mające słowo **com** w nagłówku swojego podręcznika.



# Podstawowe komendy – grep

- **Grep** służy do wyszukiwania danego ciągu znaków w podanych plikach. Podstawowym użyciem jest: `grep "wyszukiwana fraza" plik`.
- W takim przypadku wynikiem są wszystkie linie zawierające daną frazę w podanym pliku.

## Przykład

```
grep -i "lorem" readme.txt
```

## Podstawowe opcje komendy grep

<b>-i</b>	Wyszukuje, nie zważając na wielkość znaków
<b>-w</b>	Wyszukuje tylko pełne słowo
<b>-A &lt;n&gt;</b>	Pokazuje n linii po wyszukiwanym słowie
<b>-B &lt;n&gt;</b>	Pokazuje n linii przed wyszukiwanym słowem
<b>-r</b>	Wyszukuje rekursywnie we wszystkich plikach podanego katalogu
<b>-v</b>	Wyszukuje wszystkie linijki niezawierające podanego słowa
<b>-c</b>	Podaje liczbę wystąpień wyszukiwanego słowa
<b>-l</b>	Wypisuje nazwy plików, w których znalazł dane słowo
<b>-n</b>	Dodaje numer linii, w której znalazł słowo



# Podstawowe komendy

## find

Komenda wyszukująca pliki to **find**.  
Jej ogólna forma to:

**find <katalog startowy> <kryteria  
wyszukania i akcje>**

### Przykład

**find . -name "pattern" -print**



# Podstawowe opcje komendy find

<b>-atime</b> n	Plik, który został utworzony n dni temu, np. +7 – utworzony dawniej niż siedem dni temu
<b>-mtime</b> n	Plik, który został zmodyfikowany n dni temu, np. -5 – zmodyfikowany nie później niż 5 dni temu
<b>-size</b> n	Plik ma n bloków wielkości (blok to 512 bajtów), np. +100 – plik większy niż 100 bloków = 50 KB
<b>-type</b> f	Wyszukuje po typie pliku, np. f = plik tekstowy (w przykładzie), d = katalog
<b>-name</b> nam	Nazwa pliku to <b>nam</b>

<b>-user</b> usr	Nazwa właściciela pliku to usr
<b>-group</b> grp	Właściciel pliku należy do grupy grp
<b>-perm</b> p	Tryb dostępu pliku to p (gdzie p to liczba)
<b>-print</b>	Wyświetla ścieżkę do pliku
<b>-exec</b> cmd	Wykonuje komendę cmd na pliku

# Potok

## Potok

**Potok** (pipe) – jeden z mechanizmów komunikacji międzyprocesowej, umożliwiający wymianę danych pomiędzy dwoma procesami. Odbywa się to najczęściej przez połączenie STDOUT z STDIN innego procesu, na przykład:

```
ps aux | less
```

```
cat plik | grep -i a
```

<b>command &gt; file</b>	Przekierowuje STDOUT z komendy command do pliku file (nadpisując go)
<b>command &gt;&gt; file</b>	Przekierowuje STDOUT z komendy command do pliku file (rozszerzając go)
<b>command &lt; file</b>	Przekierowuje STDIN z pliku file do komendy command pliku file
<b>cat file1 file2 &gt; file0</b>	Skleja file1 i file2 wynik, zapisując do file0



**Użytkownicy**

# Rodzaje użytkowników w systemach Unix

Oto trzy główne typy użytkowników:

- **root** – tak zwany **superuser** – ma całkowity dostęp do maszyny, może wywoływać każdą komendę,
- **konta systemowe** – potrzebne do działania systemu i krytycznych dla niego procesów,
- **konta użytkowników** – konto normalnego użytkownika.

## Podstawowe komendy w systemie Linux

<b>adduser</b> username	Dodaje użytkownika do systemu	<b>-d homedir</b> – wskazuje na (już istniejący) katalog domowy <b>-g groupname</b> – dodaje do danej grupy podczas tworzenia <b>-m</b> – tworzy nowy katalog domowy <b>-l</b> – zmienia nazwę użytkownika (tylko dla usermod)
<b>usermod</b> username	Zmienia atrybuty użytkownika	
<b>passwd</b> username	Zmienia hasło użytkownika	
<b>deluser</b> username	Usuwa użytkownika	<b>-r</b> – niszczy katalog domowy danego użytkownika



# Podstawowe komendy

## Grupy

<b>addgroup</b> groupname	Dodaje grupę do systemu	<b>-g ID</b> – numer ID grupy,
<b>groupmod</b> groupname	Zmienia opcje grupy	<b>-o</b> – daje możliwość użycia zajętego już numeru ID,  <b>-r</b> – dodaje konto systemowe do grupy,  <b>-f</b> – opcja ta powoduje, że funkcja zwróci success, jeżeli grupa już istnieje,  <b>-n</b> – zmienia nazwę grupy (tylko groupmod).
<b>delgroup</b> groupname	Usuwa grupę	

## SUDO

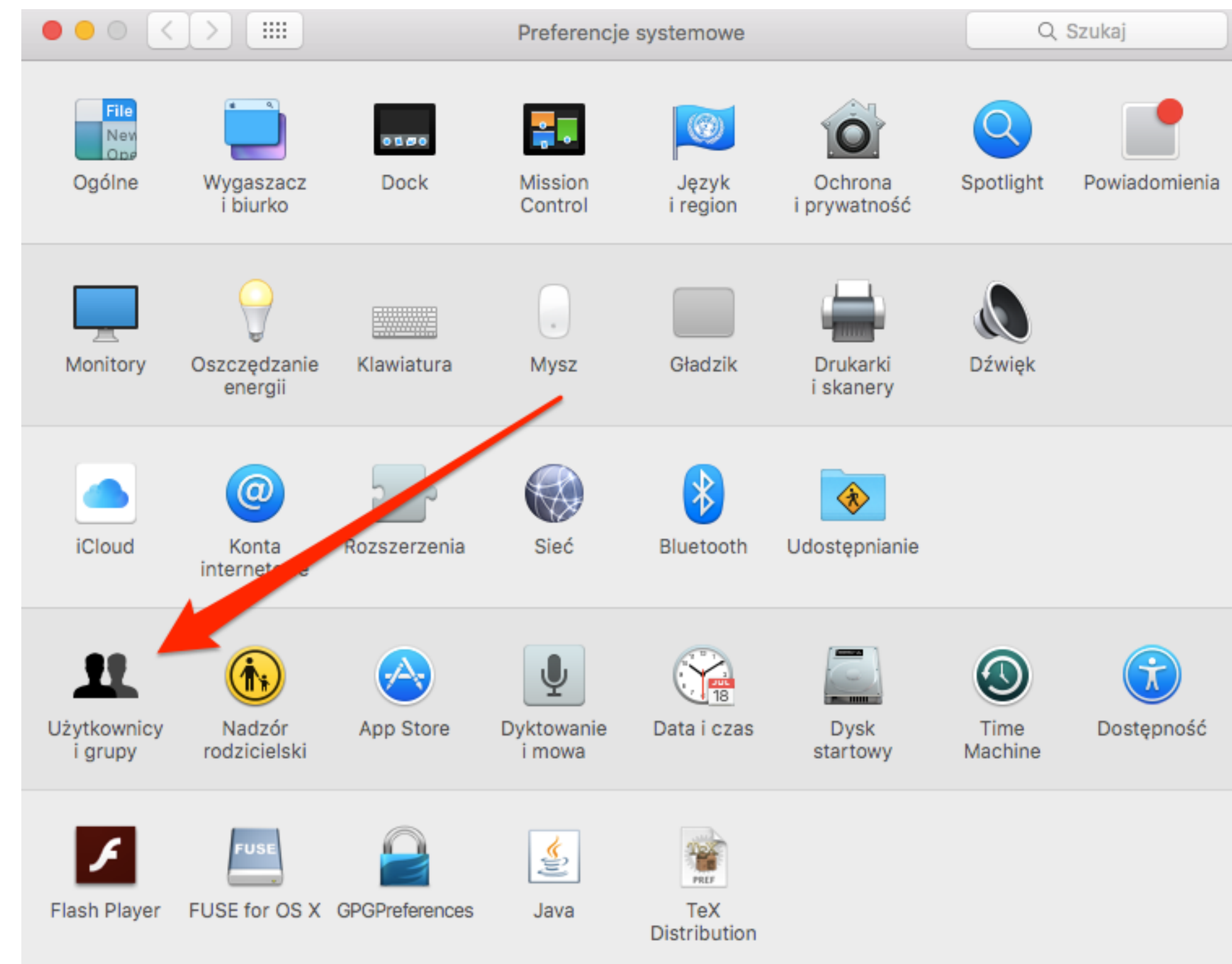
- **sudo nazwa\_komendy** – wywołuje daną komendę na poziomie administratora systemu (podobne możliwości jak **root**).
- **sudo su** – otwiera nową powłokę, w której działamy jako admin.
- **visudo** – pozwala na bezpieczną konfigurację pliku sudoers (oznaczającego, kto ma prawa do używania komendy sudo).



# Zarządzanie użytkownikami w systemie Mac

W systemie Mac zarządzanie użytkownikami odbywa się przez dedykowany ekran w preferencjach systemowych nazywany „**użytkownicy i grupy**”.

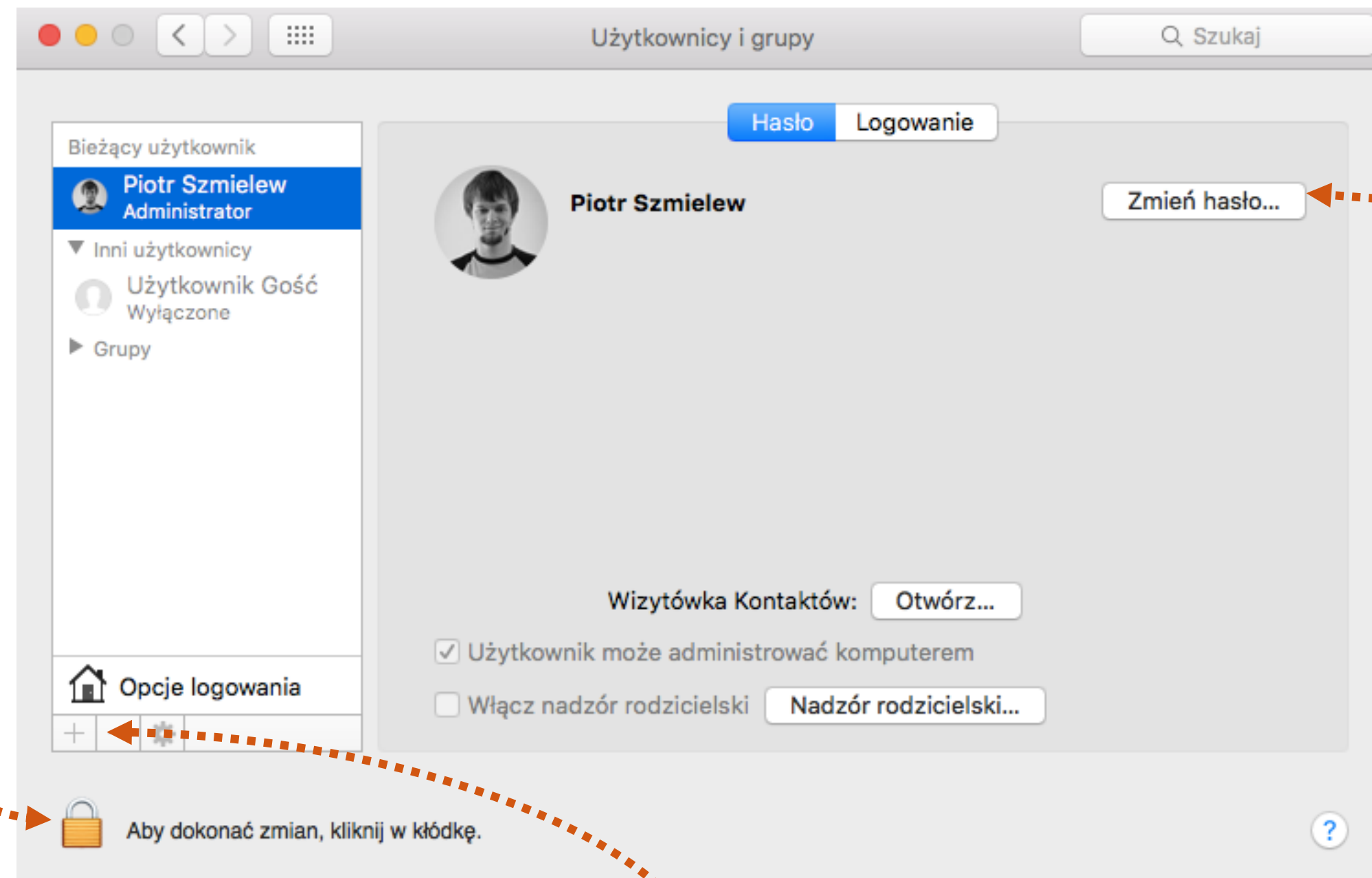
Oprócz tego polecenia **sudo** i **visudo** działają dokładnie tak samo jak w systemie Linux.



# Edycja użytkowników w systemie Mac

Najpierw odblokuj  
możliwość edycji  
użytkowników

(podaj w następnym  
okienku swoje hasło)



Możesz również zmienić  
swoje hasło

Potem możesz już dodać  
użytkownika (lub grupę) klikając znak +



**Dostęp do plików**

# Prawa dostępu

Po wpisaniu komendy:

**ls -lg**

– przykładowy output jest następujący:

**drwxr-xr-x 1 Agata 197610 0 mar 18 13:19**  
**katalog\_z\_obrazkami/**

- Pierwszy symbol (w tym zapisie **drwxr-xr-x**) oznacza, czy dany element jest katalogiem czy nie (czyli d oznacza katalog, plik jest określany kreską -)
- Następne 9, to opis praw dostępu.
- Dalej jest suma kontrolna, nazwa grupy, do której należy plik, wielkość, data utworzenia i nazwa.

## Opis rwx

- Pierwsze trzy znaki oznaczają możliwości dostępu dla właśnie zalogowanego użytkownika (r – read, w – write, x – execute)
- Dalsze trzy oznaczają dostęp dla grupy, do której należy dany plik.
- Ostatnie trzy – prawa dostępu dla wszystkich innych.

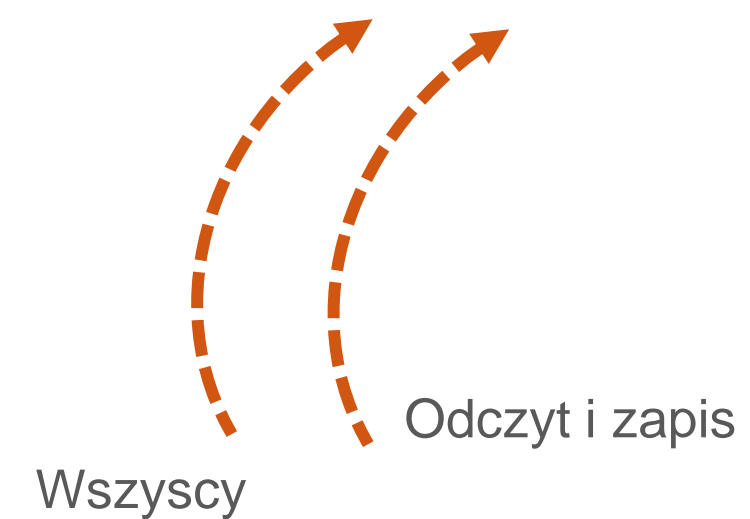
**-rwxrw-r--**

# Zmiana praw dostępu do pliku

**chmod** – komenda zmieniająca uprawnienia dostępu do pliku.

## Przykład

**chmod a=rw file.txt**



<b>U</b>	Użytkownik
<b>G</b>	Grupa
<b>O</b>	Inni
<b>a</b>	Wszyscy (to samo co połączenie <b>u</b> , <b>g</b> , <b>o</b> )
<b>r</b>	Odczyt
<b>w</b>	Zapis (i usunięcie)
<b>x</b>	Uruchomienie (w przypadku katalogu dostęp)
<b>+</b>	Dodanie uprawnień
<b>-</b>	Zabranie uprawnień

# Zmiana grupy, do której należy plik

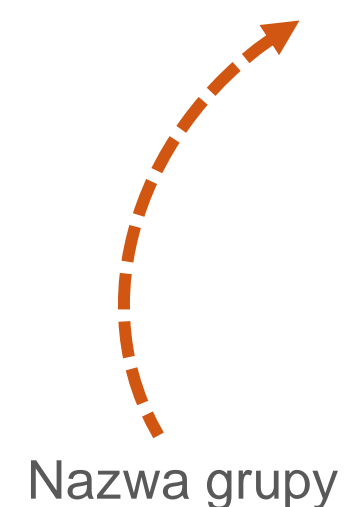
## chgrp

**chgrp** – komenda zmieniająca grupę pliku tylko do takiej grupy, do której użytkownik sam należy.

**chgrp** nazwa\_grupy plik1 plik2

### Przykład

**chgrp** CodersLab cwiczenie1.txt



Nazwa grupy

## chown

**chown** – komenda służąca do zmiany właściciela pliku (co zmienia też grupę). Może być wywoływana tylko przez administratora systemu (poprzez **sudo**).

**chown** :nazwa\_grupy plik1 plik2



Nowy właściciel pliku

Nazwa grupy, do której przypisujemy plik

Nazwa pliku, plików



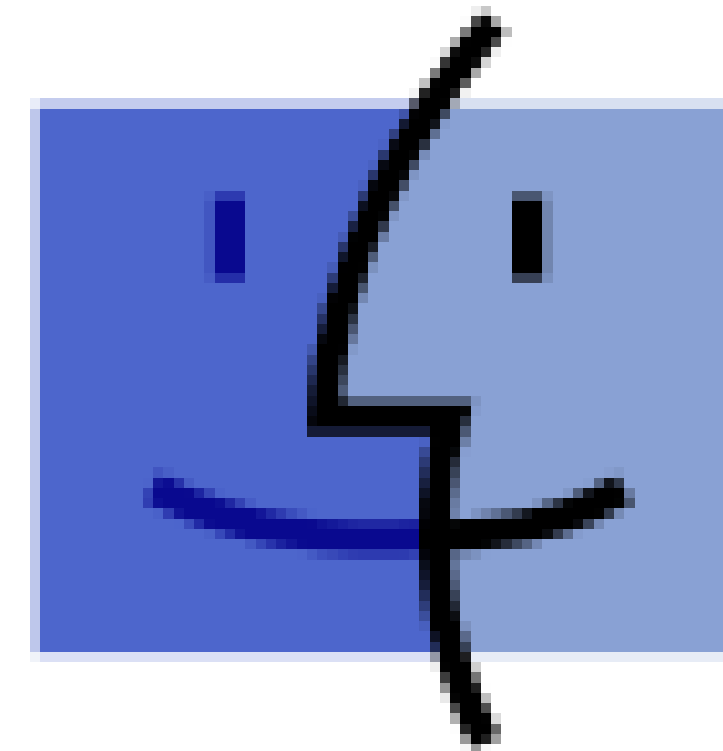


# Instalowanie nowych programów

# Instalowanie menedżera pakietów na MacOS

- System operacyjny Mac OS nie ma domyślnie menedżera pakietów.
- Aby zainstalować najpopularniejszego managera wejdź na stronę <http://brew.sh> i wpisać w terminal podaną tam komendę instalacyjną.

**Uwaga! Brew nigdy nie powinno być używane z sudo!**



Mac OS



# Zarządzanie pakietami

## Zarządzanie pakietami (apt)

Aktualizowanie listy pakietów:

- Linux  
**sudo apt-get update**
- MacOS  
**brew upgrade**

Instalacja pakietu:

- Linux  
**sudo apt-get install nazwa\_pakietu**
- MacOS  
**brew install nazwa\_pakietu**

Kasowanie pakietów:

- Linux:  
**sudo apt-get remove nazwa\_pakietu**
- MacOS:  
**brew uninstall nazwa\_pakietu**

Kasowanie pakietu z zależnościami:

- Linux:  
**sudo apt-get --purge remove nazwa\_pakietu**
- MacOS:  
**brew uninstall nazwa\_pakietu**

# Zarządzanie pakietami

Pobieranie kodów źródłowych:

- Linux  
**sudo apt-get source** nazwa\_pakietu
- MacOS – niezaimplementowane

Wyszukiwanie pakietów:

- Linux  
**sudo apt-cache search** nazwa\_pakietu
- MacOS  
**brew search** nazwa\_pakietu

# Zarządzanie pakietami

## Zarządzanie pakietami (dpkg)

- Aktualizowanie wszystkich pakietów:  
**sudo apt-get upgrade**
- Aktualizowanie dystrybucji:  
**sudo apt-get dist-upgrade**
- Kasowanie wszystkich pobranych plików:  
**sudo apt-get clean**

Polecenie **dpkg** służy do instalacji pobranych plików .deb.

- Instalacja pakietu:  
**sudo dpkg nazwa\_pakietu**
- Kasowanie pakietu:  
**sudo dpkg -r nazwa\_pakietu**



# **Tematy zaawansowane**

# Procesy

- **ps** – komenda wypisująca wszystkie procesy.
- Użyteczna w połączeniu z **grep**, poniższa komenda pokaże wszystkie procesy, które w nazwie mają „chrome”:

**ps aux | grep chrome**

- **pstree** – pokazuje procesy (tylko te należące do użytkownika) w formie drzewa procesów.

## Opcje komendy ps

<b>-a</b>	Pokazuje procesy innych użytkowników
<b>-e</b>	Pokazuje rozszerzone informacje
<b>-u</b>	Pokazuje dodatkowe informacje (jak opcja <b>-f</b> )
<b>-x</b>	Pokazuje informacje o procesach nieznajdujących się w terminalu

# Procesy

## Informacje wyświetlane przez PS

<b>UID</b>	ID użytkownika, który stworzył proces
<b>PID</b>	ID procesu
<b>PPID</b>	ID procesu rodzica
<b>C</b>	Procent CPU, jaki pochłania proces
<b>STIME</b>	Czas startu procesu
<b>TTY</b>	Terminal, na którym działa proces
<b>TIME</b>	Czas CPU, jaki proces zużył
<b>CMD</b>	Komenda, jaka wystartowała proces

# Procesy

## Typy procesów

- **Zombie** – proces, który nadal jest widoczny w tabeli procesów, choć się skończył. Stan taki może nastąpić, jeżeli proces rodzic został zamknięty niepoprawnie. Często opisywany też jako **defunct**.
- **Orphan** – działający proces, którego rodzic został zniszczony. Proces taki może cały czas poprawnie się zamknąć.
- **Deamon** – proces systemowy działający w tle bez podpiętego terminala. Zazwyczaj celem demona jest ciągle lub okresowe powtarzanie jakiegoś działania.

## Niszczanie procesów

- **kill [sygnał] [PID]** – komenda wysyłająca sygnał do procesu. Sygnały niszczące (zabijające) procesy:  
**-SIGTERM (-15)**  
**-SIGKILL (-9)**
- **killall [nazwa-procesu]** – Wysyła sygnał do wszystkich procesów o danej nazwie.
- Obie komendy wyślą **SIGTERM**, jeżeli nie zostanie podany żaden sygnał.
- Żeby zabić proces **zombie** najczęściej trzeba zabić proces jego rodzica (**PPID**).



# Praca ze zdalną konsolą

## Praca ze zdalną konsolą

- SSH – skrót od secure shell. Protokół pozwalający na bezpieczne zalogowanie się screen do komputera przez sieć.
- Logujemy się poprzez komendę:  
**ssh user@host.pl**
- Przydatne komendy podczas używania SSH:  
**w** – lista zalogowanych osób,  
**whoami** – pokazuje login aktualnie zalogowanego użytkownika,  
**uptime** – pokazuje, ile czasu upłynęło od startu systemu.

## Komenda screen

- **screen** – program pozwalający na tworzenie wirtualnych sesji. Sesje te działają do czasu wyłączenia systemu lub ręcznego ich zamknięcia. Bardzo przydatne przy uruchamianiu skryptów przez SSH.
- **screen -S nazwa\_sesji** – tworzy sesję o podanej nazwie.
- **screen -d -R nazwa\_sesji** – przywraca sesję.
- **CTRL+A+D** – odłącza sesję (nie zamykając jej).
- **CTRL+A+K** – zamyka sesję.



# Harmonogram zadań

- **cron** – demon (proces działający w tle), którego praca polega na okresowym wywoływaniu innych programów.
- **crontab** – tabela zadań, które **cron** ma uruchamiać, z dokładnym określeniem czasu, w którym mają być uruchomione.

## Opcje

<b>-e</b>	Edycja.
<b>-v</b>	Wyświetlenie czasu ostatniej edycji.
<b>-l</b>	Wyświetlenie.
<b>-r</b>	Usunięcie całego pliku crontab.

# Przykładowy wygląd pliku crontab

## Przykład

- Aby dodać zadanie, które będzie uruchomione co określony czas, musimy dodać \*/<odstęp czasu> w odpowiednim polu.

- \*/5 \* \* \* \* /backup.sh  
Uruchomi skrypt co pięć minut.

```
# For details see man 4 crontabs

# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * user-name  command to be executed
```

# Zmienne systemowe

- **Zmienne środowiska shell** – zmienne krótkoterminowe, czyszczone pod koniec działania powłoki.
- **Zmienne systemowe** – zmienne długoterminowe, zapamiętywane między sesjami użytkownika.
- Wypisanie zmiennej:  
**echo \$<nazwa\_zmiennej>**
- Nastawienie zmiennej:  
**set <nazwa\_zmiennej> = wartość**

Zmienne systemowe		Zmienne shella	
USER	Nazwa zalogowanego użytkownika	cwd	Ścieżka, w której się znajdujesz
HOME	Ścieżka do katalogu home	home	Ścieżka katalogu domowego
HOST	Nazwa komputera	path	Katalogi, w których shell szuka programów do wywołania
ARCH	Architektura procesora		
DISPLAY	Nazwa środowiska graficznego		
PATH	Katalogi, w których shell szuka programów do wywołania		

# Symlinki i hardlinki

## Tworzenie

- **symlink** – wskaźnik na plik znajdujący się w innym miejscu. Jeżeli zmienimy nazwę pliku lub przeniesiemy go, **symlink** zostanie zepsuty. Jeżeli plik zostanie podmieniony, **symlink** zacznie wskazywać na nowy plik.
- **hardlink** – wskaźnik na docelowe miejsce na dysku (**inode**). W chwili przeniesienia pliku **hardlink** będzie poprawnie na niego wskazywał. Może być utworzony tylko na tym samym systemie plików.

- **Symlinki:**  
In /root/file1 /root/file2
- **Hardlinki:**  
In -s /root/file1 /root/file2



# Różnorodność w systemach Linux



# Najpopularniejsze wersje Linuksa

## ➤ Ubuntu

- jedna z najpopularniejszych dystrybucji Linuksa,
- ma wiele własnych dystrybucji.

## ➤ Linux Mint

- user experience bardzo podobny do systemu Windows,
- system działający na zasadzie out of the box.

## ➤ Debian

- jedna ze starszych dystrybucji,
- służył jako baza m.in. dla Ubuntu,
- czysty system operacyjny.

## ➤ Fedora

- system wprowadzający najwięcej zmian, ciągle dodający najnowsze udogodnienia,
- bardziej problematyczna instalacja systemu, mniejsza stabilność.

## ➤ OpenSUSE

- alternatywa dla Mint, Ubuntu i podobnych systemów,
- łatwy w instalacji i użytkowaniu.

## ➤ Arch

- system dla zaawansowanych użytkowników,
- Daje możliwość stworzenia całkowicie spersonalizowanego systemu.

# Najpopularniejsze typy shelli

## Bourne Shell (sh)

- Dostępna na każdym systemie typu Unix (wyznacza standard).
- Druga powłoka używana w systemach Unix (stworzona w 1977 roku).
- Główne ograniczenie to niemożliwość działania na liczbach całkowitych bez tworzenia nowego procesu.
- Można go zidentyfikować podczas używania po znaku \$ znajdującym się na początku linii.

## Bash

- Akronim od Bourne-Again Shell.
- Domyślna powłoka w większości systemów typu Linux oraz w systemie Mac OS X (wersie 10.3+).
- Pozwala na pracę w trybie konwersacyjnym (interaktywne wprowadzanie poleceń) i wsadowym (poprzez skrypty).
- Rozszerza standard **sh** np. przez:
  - działania na liczbach całkowitych,
  - przekierowywanie wejścia i wyjścia,
  - wyrażenia regularne (Bash 3.0+).

# Najpopularniejsze typy shelli

## Z shell (zsh)

- Potężne rozwinięcie standardu **sh** dla zaawansowanych użytkowników zawierające m.in.:
  - programowalne autouzupełnianie komend,
  - współdzielenie historii komend pomiędzy działającymi powłokami,
  - rozbudowane wyszukiwanie plików (nieopierające się na programach typu **find**),
  - autokorektę,
  - całkowitą kompatybilność z **sh** (może się podszywać pod powłokę **sh**).

## C shell (csh)

- Powłoka stworzona dla systemu BSD.
- Główna zmiana polega na stworzeniu języka podobnego do C jako języka głównego powłoki.
- Pomimo dodania wielu usprawnień do standardu powłoka nie przyjęła się i jest uważana za problematyczną.

# Najlepsze emulatory terminalu

## Terminator

- Zaawansowany i uznawany za jeden z najlepszych emulatorów.
- Główne jego funkcjonalności to:
  - różne schematy kolorystyczne (także user defined),
  - możliwość doinstalowania różnych pluginów,
  - dodatkowe skróty klawiszowe dla najczęstszych komend,
  - dzielenie okna na pomniejsze wirtualne terminale i możliwość zmiany ich wielkości.

## Guake

- Emulator całkowicie napisany w Pythonie.
- Jako jeden z pierwszych wprowadził ukrywanie emulatora pod górnym paskiem systemowym (bazowane na emulatorach z gier FPS).
- Stworzony dla środowiska graficznego GNOME.

# Najlepsze emulatory terminalu

## Yakuake

- Emulator podobny do Guake, przeznaczony dla systemów opartych na środowisku graficznym KDE.
- Główne cechy:
  - konfigurowalna wielkość i animacja opadania,
  - interfejs tabelkowy.







GADAĆ JEST ŁATWO.  
POKAŹCIE MI KOD.

L. TORVALDS