

Wstęp do systemu Linux / Mac

v3.1

Plan

- Nauka obsługi systemu Linux / terminala Mac
- Wprowadzenie do systemów Unix
- Poruszanie się po konsoli, system plików
- Użytkownicy
- Dostęp do plików
- Instalowanie nowych programów
- Tematy zaawansowane
- Różnorodność w systemach Linux

Nauka obsługi systemu Linux / terminala Mac

Obsługa systemu Linux / terminala Mac

Prawie cały Internet jest oparty o systemy typu Linux. Dlatego musisz nauczyć się chociaż podstaw korzystania z tego systemu (i to używając tylko konsoli):

Obejrzyj wideotutoriale:

Najłatwiej Ci będzie nauczyć się pracy w konsoli Linuxowej patrząc jak ktoś z niej korzysta. Dlatego obejrzyj

<https://goo.gl/C0z6KD>

Jeśli korzystasz z Maca – nie zrażaj się i również obejrzyj ten poradnik. Prawie wszystkie komendy zadziałają również u Ciebie! Informacje o różnicach znajdziesz podczas przerabiania prezentacji „Podstawy Linuxa / Mac”.

Przeczytaj prezentację i zrób ćwiczenia do niej

Wprowadzenie do systemów Unix

Wprowadzenie do systemów Unix

Krótko o Linuksie

- GNU/Linux (dalej będziemy go nazywać **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	Przerywa pracę procesu.
CTRL + D	Wysyła sygnał EOF (end-of-file).
CTRL + R	Przeszukuje historię pod względem podanych liter.
CTRL + Z	Zatrzymuje proces.
CTRL + A	Przenosi kursor 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**.

Edytor 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

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

**Poruszanie się
po konsoli,
system plików**

Struktura katalogów w systemie Linux

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

/etc/init.d	Katalog zawierający skrypty uruchamiane podczas startu systemu.
/etc/profile.d	Katalog uruchamiający skrypty uruchamiane przy logowaniu danego użytkownika.
/home	Katalog domowy użytkownika.
/root	Katalog domowy użytkownika root (głównego użytkownika systemu).
/tmp	Katalog zawierający pliki chwilowe potrzebne do działania programów i systemu.
/usr	Katalog zawierający pliki wykonywalne programów, kod źródłowy, biblioteki i dokumentacje.

Struktura katalogów w systemie Mac

/	Główny katalog w systemie (wszystkie katalogi są podkatalogami /).
/Applications	Katalog, w którym instalowane są aplikacje.
/Volumes	Katalog, w którym montowane są wszystkie dyski (w tym pliki dmg z aplikacjami) np. CD-ROM.
/etc	Katalog zawierający pliki konfiguracyjne, pliki używane przez podsystemy Maca (np. bazy danych).
/sbin	Katalog zawierający podstawowe pliki binarne potrzebne do działania systemu.
/Library	Katalog zawierający biblioteki zainstalowane w systemie.
/bin	Katalog zawierający programy.

/dev	Katalog, w którym znajdują się wszystkie urządzenia.
/etc/profile	Katalog uruchamiający skrypty uruchamiane przy logowaniu danego użytkownika.
/Users	Katalog domowy użytkownika.
/System	Katalog zawierający pliki systemowe.
/tmp	Katalog zawierający pliki chwilowe potrzebne do działania programów i systemu.
/usr	Katalog zawierający pliki wykonywalne programów, kod źródłowy, biblioteki i dokumentacje.

Podstawowe komendy – pliki i katalogi

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

Podstawowe komendy – pliki i katalogi

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

Podręczniki systemowe

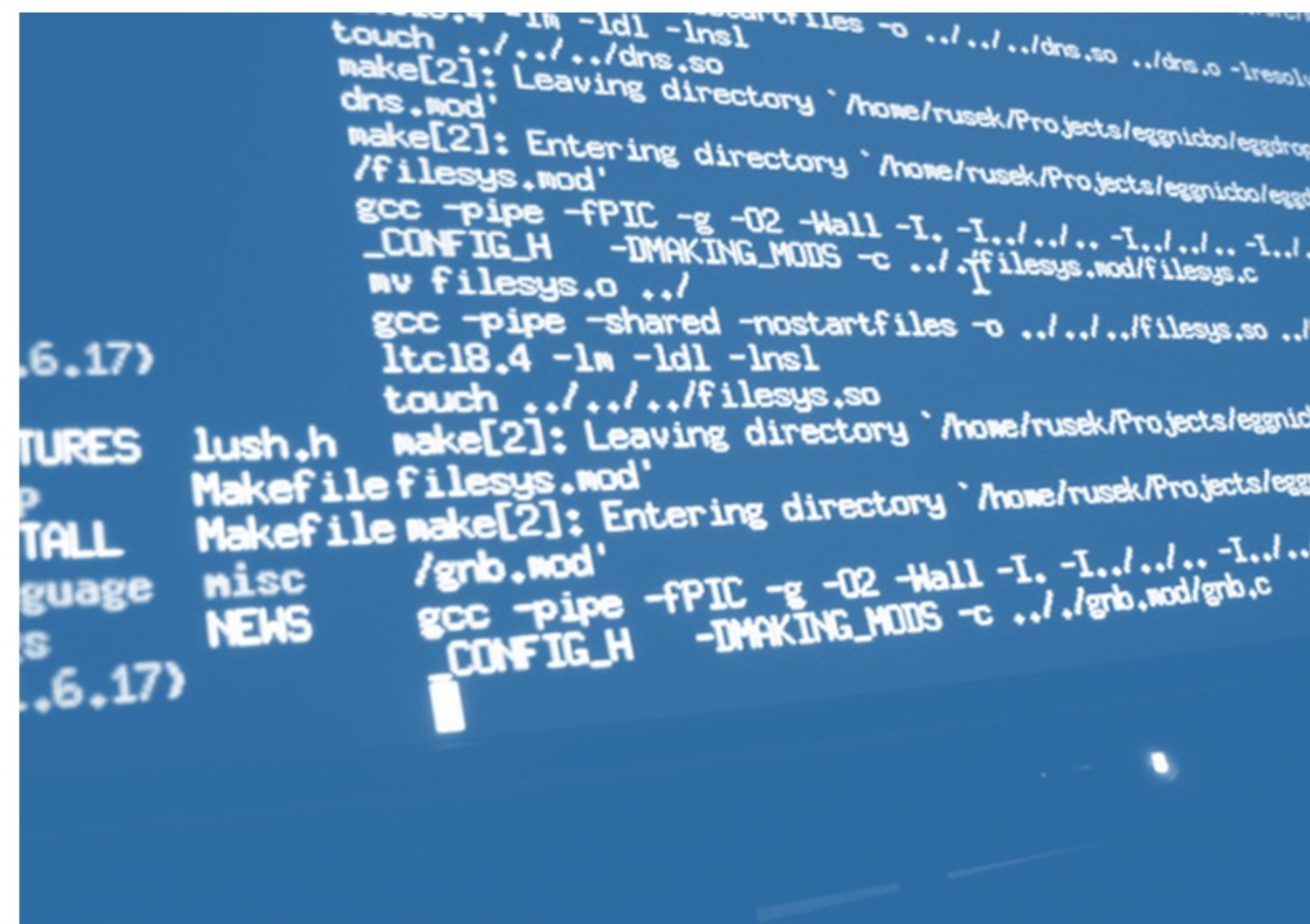
man <nazwa_komendy> – otwiera podręcznik pomocy danej komendy np. **man ls**.

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

-i	Wyszukuje, nie zważając na wielkość znaków.
-w	Wyszukuje tylko pełne słowo.
-A <n>	Pokazuje n linii po wyszukanym słowie.
-B <n>	Pokazuje n linii przed wyszukanym słowem.
-r	Wyszukuje rekursywnie we wszystkich plikach podanego katalogu.
-v	Wyszukuje wszystkie linijki niezawierające podanego słowa.
-c	Podaje liczbę linii, które pasują do wzorca.
-l	Wypisuje nazwy plików, w których znalazł dane słowo.
-n	Dodaje numer linii, w której znalazł słowo.

Podstawowe komendy

find

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

```
find <katalog_startowy> <kryteria>  
<akcje>
```

Przykład:

```
find . -name "pattern" -print
```



Podstawowe opcje komendy find

-atime n	Plik, który został otwarty n dni temu. Np. +7 – otwarty dawniej, niż 7 dni temu.
-mtime n	Plik, który został zmodyfikowany n dni temu. Np. -7 – zmodyfikowany nie później, niż 5 dni temu.
-size n	Plik ma n bloków wielkości (blok to 512 bajtów). Np. +100 – plik większy niż 100 bloków (50kB).
-type f	Wyszukuje po typie pliku. Np. f = plik tekstowy (w przykładzie), d = katalog.
-name nam	Wyszukuje plik wg nazwy (w tym przypadku nam).

-user usr	Nazwa właściciela pliku to usr .
-group grp	Właściciel pliku należy do grupy grp .
-perm p	Typ dostępu do pliku to p (gdzie p to liczba).
-print	Wyświetla ścieżkę do pliku.
-exec cmd	Wykonuję 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
```

<code>command > file</code>	Przekierowuje STDOUT z komendy command do pliku file , nadpisując go.
<code>command >> file</code>	Przekierowuje STDOUT z komendy command do pliku file , rozszerzając go.
<code>command < file</code>	Przekierowuje STDIN z pliku file do komendy command .
<code>cat file1 file2 > file0</code>	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

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

Podstawowe komendy

Grupy

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

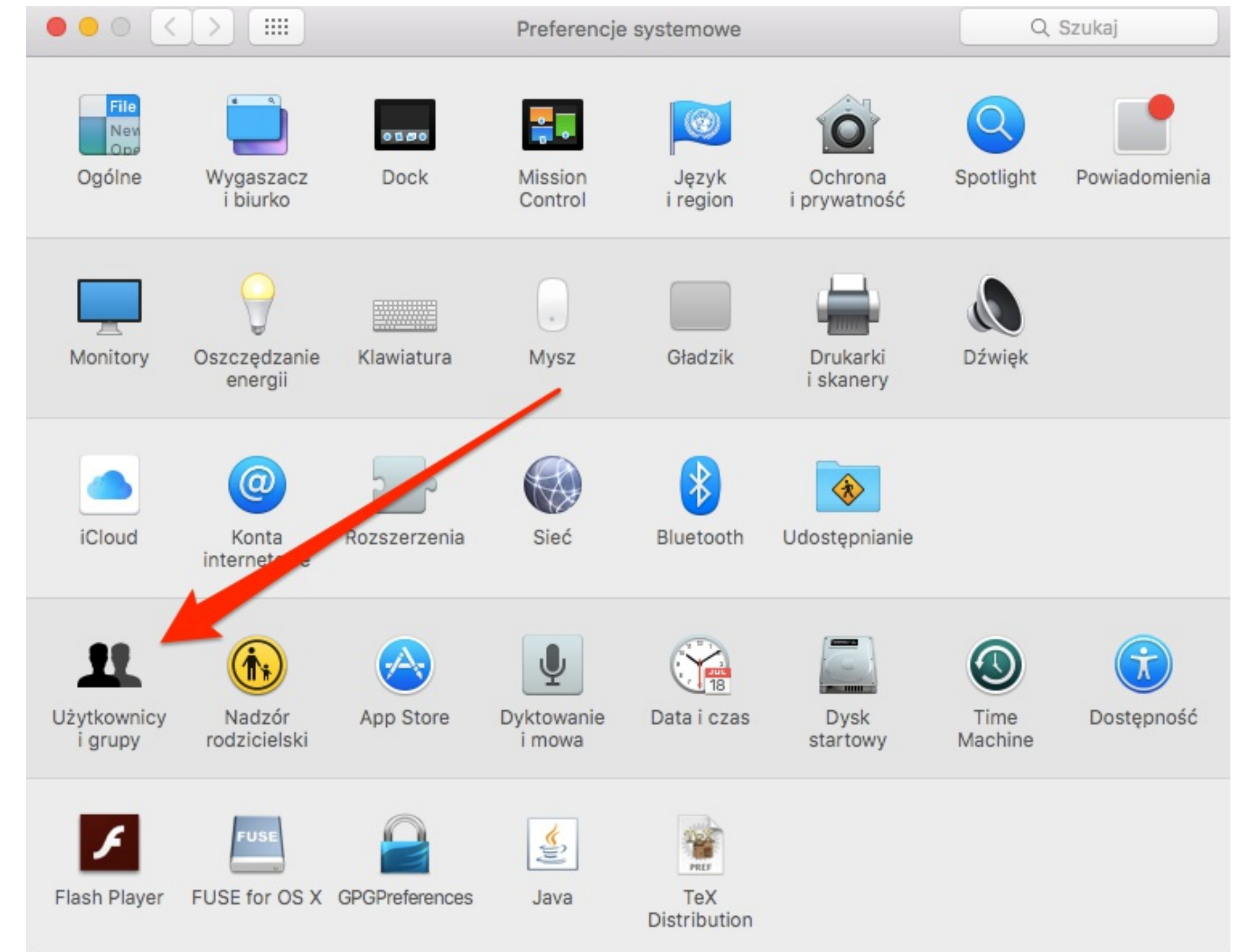
sudo

- **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.

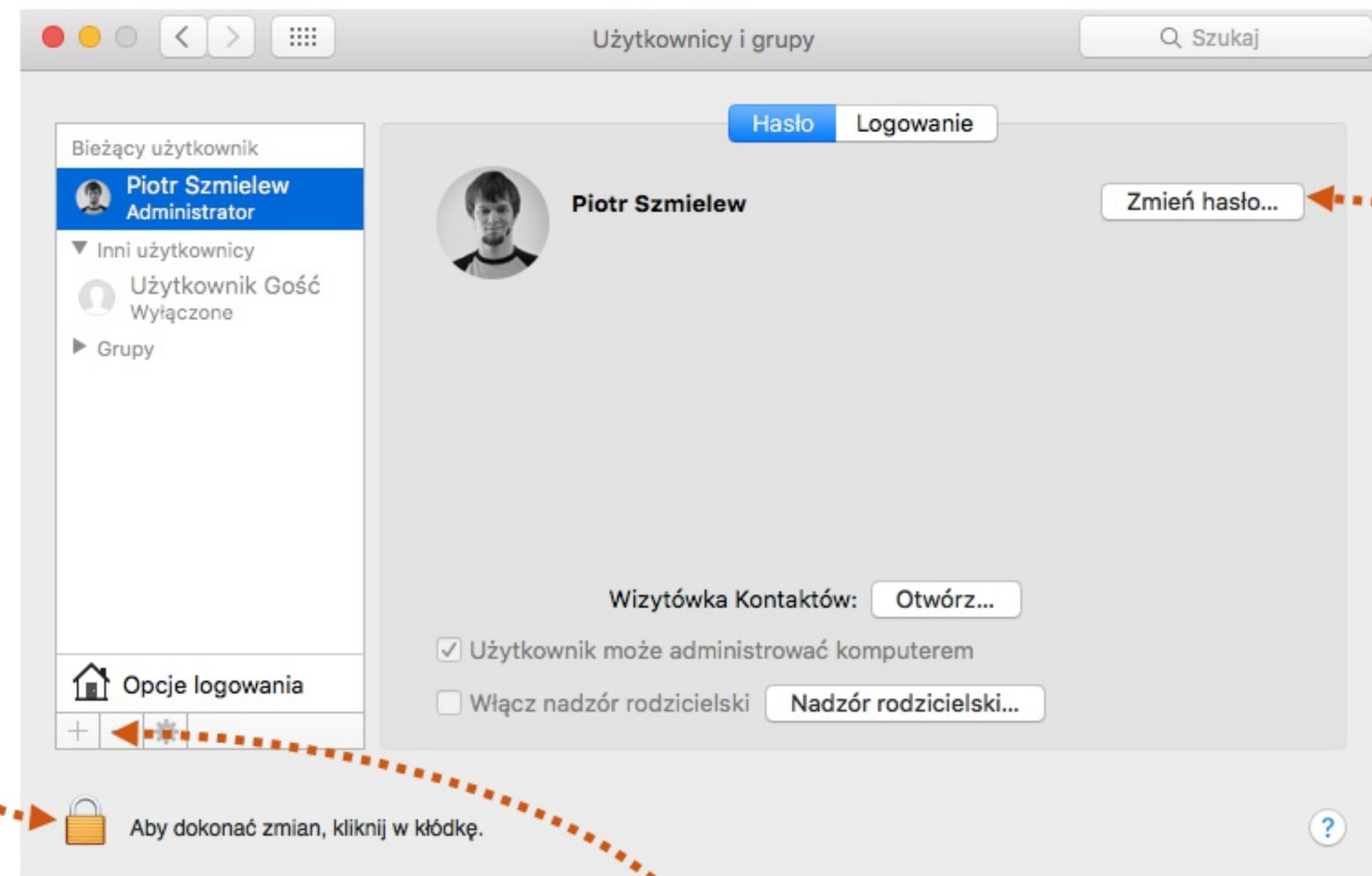


Edycja użytkowników w systemie Mac

Najpierw odblokuj
możliwość edycji
użytkowników

(podaj w następnym
okienku swoje hasło)

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



Możesz również zmienić
swoje hasło

Dostęp do plików

Prawa dostępu

Po wpisaniu komendy: `ls -lg`

– przykładowy output jest następujący:

```
drwxr-xr-x 9 Agata 4096 kwi 9 14:22 M_00_S_07_Podstawy_Linuxa_lub_Maca
```

- 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 **nazwa grupy**, do której należy plik, **wielkość**, **data utworzenia** i **nazwa pliku** (lub katalogu).

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.

-rwxrwx-r--

Zmiana praw dostępu do pliku

chmod – komenda zmieniająca uprawnienia dostępu do pliku.

Przykład:

chmod a=rw file.txt

- **a** – oznacza wszystkich użytkowników systemu,
- **rw** – oznacza **odczyt** i **zapis**,
- **file.txt** – nazwa pliku, któremu zmieniamy uprawnienia.

Zapis ten oznacza, że użytkownikom zdefiniowanym przed znakiem **=** przyporządkowujemy prawa zdefiniowane po znaku. Szczegóły w tabelce obok.

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

Zmiana grupy, do której należy plik

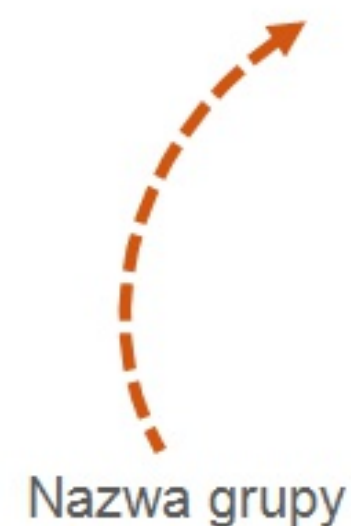
chgrp

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 username:groupname file1 file2...



Nowy właściciel pliku

Nazwa grupy, do której przypisujemy plik

Nazwa pliku, plików

Instalowanie nowych programów

Instalowanie menedżera pakietów na MacOS

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

Uwaga! Brew nigdy nie powinno być używane z sudo!

Jeśli użyłeś naszego programu instalacyjnego będziesz już mieć zainstalowane Homebrew.



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 pakietu:

- Linux
**sudo apt-cache search
nazwa_pakietu**
- MacOS
brew search nazwa_pakietu

Aktualizowanie wszystkich pakietów:

sudo apt-get upgrade

Aktualizowanie dystrybucji:

sudo apt-get dist-upgrade

Kasowanie wszystkich pobranych plików:

sudo apt-get clean

Zarządzanie pakietami

Zarządzanie pakietami (Debian, Ubuntu, Mint)

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

- Instalacja pakietu:
sudo dpkg -i 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

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

Procesy

Informacje wyświetlane przez PS

UID	ID użytkownika, który stworzył proces
PID	ID procesu
PPID	ID procesu rodzica
C	Procent CPU, jaki pochłania proces
STIME	Czas startu procesu
TTY	Terminal, na którym działa proces
TIME	Czas CPU, jaki proces zużył
CMD	Komenda, która 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 ciągłe 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 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

-e	edycja
-v	wyświetlanie czasu ostatniej edycji
-l	wyświetlenie
-r	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 domowego	home	Ścieżka katalogu domowego
HOST	Nazwa komputera	path	Katalogi, w których shell szuka programu do wywołania.
ARCH	Architektura procesora		
DISPLAY	Nazwa środowiska graficznego		
PATH	Katalogi, w których shell szuka programów do wywołania.		

Symlinki i hardlinki

- **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.

Tworzenie

- hardlink:

```
ln /root/file1 /root/file2
```

- symlink:

```
ln -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.

Najpopularniejsze wersje Linuksa

➤ 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