Materiały do laboratoriów

Spis treści

[12pt,a4paper]article
[polish]babel [utf8]inputenc [T1]fontenc geometry setspace
margin=2.5cm
Materiały do laboratoriów

Spis treści

[a4paper, 12pt]article

[utf8]inputenc [T1]fontenc [polish]babel geometry xcolor hyperref graphicx fancyhdr beramono

minted [most]tcolorbox minted,breakable

a4paper, left=2.5cm, right=2.5cm, top=2.5cm, bottom=2.5cm titlebgHTML4A90E2

colorlinks=true, linkcolor=titlebg, urlcolor=titlebg, citecolor=titlebg,

[L]Wprowadzenie do pracy w systemie Linux [C]3

definitionbox[1] colback=black!5, colframe=titlebg, fonttitle=**, coltitle=white, colbacktitle=titlebg, title=1, boxrule=1pt, arc=4mm, bottomrule=0.5pt, toprule=0.5pt, breakable**

importantbox[1] colback=yellow!10, colframe=orange!80!black, fonttitle=**, title=1, boxrule=1pt, arc=2mm, breakable**

terminal listing only, listing engine=minted, minted language=bash, minted style=vs, minted options= autogobble, breaklines, fontsize=, colback=black!70, colframe=black!50, breakable, arc=3mm, boxrule=1pt, left=5mm, top=3mm, bottom=3mm, title=[on line, size=small, boxrule=0pt, colback=red!80!black, arc=3mm] [on line, size=small, boxrule=0pt, colback=yellow!80!black, arc=3mm] [on line, size=small, boxrule=0pt, colback=green!70!black, arc=3mm] **Terminal – Bash** ,

Wprowadzenie do pracy w środowisku Linux

Notatki z zajęć laboratoryjnych CZĘŚĆ 1

Spis treści

Wprowadzenie

Niniejszy dokument stanowi wprowadzenie do podstaw systemu operacyjnego Linux oraz pracy w jego interfejsie wiersza poleceń. Interfejs ten, zwany powłoką, w większości systemów Linux jest realizowany przez program https://www.gnu.org/software/bash/**Bash**. Celem jest przedstawienie fundamentalnych koncepcji teoretycznych, które odróżniają Linuksa od innych systemów, a następnie zapoznanie z podstawowymi poleceniami niezbędnymi do wydajnej pracy w środowisku tekstowym.

1 Podstawy teoretyczne: System operacyjny

1.1 Czym jest system operacyjny?

System Operacyjny (OS) Jest to nadrzędne oprogramowanie, które zarządza wszystkimi zasobami komputera. Działa jako pośrednik między użytkownikiem a sprzętem (procesorem, pamięcią, dyskami). Umożliwia uruchamianie programów i wykonywanie zadań bez konieczności znajomości technicznych detali działania podzespołów. Najpopularniejsze systemy to Windows, Linux, macOS i Android.

1.2 Serce systemu, czyli jądro (kernel)

Jądro systemu (ang. kernel) To centralna i najważniejsza część systemu operacyjnego. Odpowiada za fundamentalne zadania: zarządzanie procesami (uruchomionymi programami), alokację pamięci RAM oraz komunikację ze wszystkimi urządzeniami podłączonymi do komputera. Można je postrzegać jako "mózg"operacji systemowych.

1.3 Dwa światy: Linux vs. Windows

Chociaż oba systemy służą do zarządzania komputerem, ich filozofia i budowa znacząco się różnią.

1.3.1 Porównanie jąder systemowych

- **Linux** wykorzystuje jądro **monolityczne**. Oznacza to, że większość kluczowych funkcji jest zintegrowana w jednym, dużym programie. Jądro Linuksa jest otwarte (*open-source*), co oznacza, że każdy może przeglądać i modyfikować jego kod. Jest też modularne – sterowniki można dodawać i usuwać w trakcie pracy systemu.
- **Windows** wykorzystuje jądro **hybrydowe** (o nazwie *NT Kernel*). Łączy ono cechy jądra monolitycznego (dla wydajności) z mikrojądrem (dla stabilności). Kod jądra Windows jest zamknięty i stanowi własność firmy Microsoft.

1.3.2 Zalety i wady obu systemów

[raster columns=2, raster equal height, colback=black!5, colframe=titlebg, fonttitle=**] [title=Windows] Zalety:**

- **Prostota obsługi i intuicyjność.**
- **Ogromna kompatybilność z oprogramowaniem i grami.**

Wady:

- **System jest płatny (licencja).**
- **Postrzegany jako bardziej podatny na wirusy.**

[title=Linux] Zalety:

- **Jest darmowy i otwartoźródłowy.**
- **Wysoka stabilność i bezpieczeństwo.**

Wady:

- **Wyższy próg wejścia dla początkujących.**
- **Mniejsza liczba komercyjnych programów i gier.**

1.3.3 Przykłady wykorzystania

Windows jest idealnym wyborem dla użytkowników domowych i biurowych, którzy potrzebują systemu działającego "od razu"do przeglądania internetu, pracy z dokumentami czy rozrywki.

Linux jest narzędziem dla osób, które potrzebują pełnej kontroli nad środowiskiem pracy – programistów, administratorów serwerów czy naukowców. Umożliwia precyzyjną konfigurację każdego elementu systemu.

1.4 Świat Linuksa: Dystrybucje

Linux nie jest jednym, konkretnym systemem, lecz jądrem, na bazie którego tworzone są tzw. **dystrybucje**. Są to kompletne systemy operacyjne z jądrem Linux oraz zestawem programów.

- https://www.debian.org/**Debian**: Znany ze swojej stabilności, często używany na serwerach.
- https://ubuntu.com/**Ubuntu**: Bardzo popularny, uważany za przyjazny dla początkujących.

- https://linuxmint.com/**Linux Mint**: Ceniony za elegancki interfejs, ułatwiający przejście z Windowsa.
- **Hannah Montana Linux**: Przykład, że na bazie Linuksa można stworzyć niemal wszystko.

2 Podstawowe komendy w systemie Linux

2.1 Powłoka systemowa: Bash

Zanim przejdziemy do poleceń, warto zrozumieć, gdzie je wpisujemy. Terminal to program, który emuluje tekstowy interfejs, a wewnątrz niego działa **powłoka systemowa** (ang. *shell*). Powłoka systemowa (shell) To program-interpreter, który tłumaczy polecenia wpisywane przez użytkownika na język zrozumiały dla jądra systemu operacyjnego. Jest to podstawowe narzędzie do interakcji z systemem w trybie tekstowym. Najpopularniejszą powłoką w świecie Linuksa jest **Bash** (Bourne-Again SHell). Bash oferuje wiele ułatwień, takich jak historia wpisywanych poleceń (dostępna strzałkami w górę/dół), autouzupełnianie (klawisz `Tab`) oraz możliwość tworzenia skryptów automatyzujących zadania.

2.2 Struktura poleceń: opcje i argumenty

Praca w terminalu polega na wpisywaniu poleceń według określonego schematu:

```
polecenie [opcje] [argumenty]
```

- **Polecenie** to nazwa programu, który chcemy uruchomić (np. 'ls', 'cp').
- **Opcje** (nazywane też flagami lub przełącznikami) to modyfikatory, które zmieniają domyślne zachowanie polecenia. Zwykle poprzedzone są myślnikiem.
- **Argumenty** to obiekty, na których polecenie ma operować (np. nazwy plików, ścieżki do katalogów).

Opcje występują w dwóch formach:

- **Krótka forma**: jeden myślnik i jedna litera, np. `-l`. Krótkie opcje można łączyć. Zamiast pisać `ls -l -h -a`, można to skrócić do `ls -lha`.
- **Długa forma**: dwa myślniki i pełna nazwa, np. `-list`. Są bardziej czytelne, ale nie można ich łączyć. Pełny odpowiednik `ls -lha` to `ls -list -human-readable -all`.

Instrukcja obsługi: polecenie `man` Jeśli nie wiesz, jak działa polecenie lub jakich opcji można użyć, skorzystaj z wbudowanej instrukcji (manuala). Wpisz `man <nazwa_polecenia>`, np. `man ls`, aby wyświetlić jego pełną dokumentację. Z manuala wychodzi się, wciskając klawisz `q`.

2.3 Polecenia informacyjne

Poniżej znajdują się podstawowe polecenia służące do uzyskiwania informacji o systemie i użytkowniku.

Polecenie `echo`

Wyświetla tekst (argument) na standardowym wyjściu (ekranie). Wyświetla podany tekst na ekranie $echo'HelloWorld'$

Polecenie `whoami`

Wyświetla nazwę aktualnie zalogowanego użytkownika. Wyświetla nazwę zalogowanego użytkownika *whoami*

Polecenie `groups`

Pokazuje nazwy grup, do których należy bieżący użytkownik. Pokazuje grupy, do których należy użytkownik *groups*

Polecenie `date`

Wyświetla aktualną datę i godzinę. Jego format można kontrolować. Wyświetla bieżącą datę i godzinę *date*

Wyświetla datę w formacie ROK-MIESIĄC-DZIEŃ $date + ”$

Polecenie `uname`

Wyświetla informacje o systemie operacyjnym i jego jądrze. Wyświetla informacje o jądrze systemu (-a to –all) $uname - a$

2.4 Nawigacja i listowanie plików

Te polecenia są kluczowe do poruszania się po systemie plików.

Polecenie `pwd`

Drukuje pełną ścieżkę do bieżącego katalogu roboczego (print working directory). Sprawdź, gdzie jesteś *pwd*

Polecenie `cd`

Zmienia bieżący katalog (change directory). Zmień katalog na podany $cd/sciezka/do/katalogu$

Przejdź do katalogu domowego użytkownika *cd*

Wróć do poprzedniego katalogu $cd-$

Polecenie `ls`

Wyświetla listę plików i katalogów w bieżącej lokalizacji. Wyświetl zawartość katalogu (list) ls

Opcje dla 'ls': -l: format listy (szczegółowy) -h: rozmiary w czytelnej formie (human-readable) -a: pokaż wszystkie pliki, także ukryte (zaczynające się od .) -t: sortuj według czasu modyfikacji (najnowsze pierwsze) $ls - lhat$

2.5 Zarządzanie plikami i katalogami

Polecenie `touch`

Tworzy nowy, pusty plik lub aktualizuje datę modyfikacji istniejącego pliku. Utwórz pusty plik $touchnowy_plik.txt$

Polecenie `mkdir`

Tworzy nowy katalog (make directory). Utwórz nowy katalog $mkdirnowy_katalog$

Utwórz całą ścieżkę katalogów (parents) $mkdir - pA/B/C$

Polecenie `cp`

Kopiuje pliki lub katalogi. Kopiuj plik (cp <źródło> <cel>) -v: tryb gadatliwy (verbose), pokazuje co robi -i: tryb interaktywny, pyta przed nadpisaniem $cp{-}ivplik.txt/tmp/kopia_pliku.txt$

Kopiuj cały katalog (opcja -r, recursive) $cp - rmoj_folder//tmp/$

Polecenie `mv`

Przenosi lub zmienia nazwę plików i katalogów (move). Zmień nazwę pliku (mv <stara> <nowa>) $mvplik.txtdokument.txt$

Przenieś plik do innego katalogu z trybem gadatliwym $mv - vdokument.txt/tmp/$

Polecenie `rm`

Usuwa pliki lub katalogi (remove). Usuń plik $rmplik.txt$

Usuń katalog i całą jego zawartość -r: rekurencyjnie (dla katalogów) -f: wymuszenie (force), nie pyta o potwierdzenie $rm - rfstary_katalog/$

UWAGA! Polecenie `rm -rf` jest ekstremalnie niebezpieczne. Usuwa wszystko bezpowrotnie i bez pytania o potwierdzenie. Używaj go z najwyższą ostrożnością, zawsze upewniając się, w którym katalogu się znajdujesz (`pwd`).

2.6 Przeglądanie i przetwarzanie plików

Polecenie `cat`

Wyświetla zawartość pliku na standardowym wyjściu. Wyświetl całą zawartość pliku z numerami linii (-n) $cat - nskrypt.sh$

Polecenie `head`

Wyświetla pierwsze linie pliku (domyślnie 10). Wyświetl pierwsze 5 linii pliku $head - n5/var/log/syslog$

Polecenie `tail`

Wyświetla ostatnie linie pliku (domyślnie 10). Wyświetl ostatnie 3 linie pliku $tail - n3/var/log/syslog$

Śledź plik na żywo (idealne do logów, -f to follow) $tail - f/var/log/syslog$

Polecenie `wget`

Pobiera pliki z internetu. Pobierz plik i zapisz pod inną nazwą (-O) $wget - Opan_tadeusz.txt"https://wolnelektury.pl/media/book/txt/pan - tadeusz.txt"$

Przekierowanie wyjścia >

Zapisuje wynik (standardowe wyjście) polecenia do pliku, nadpisując jego zawartość, jeśli plik istnieje. Zapisz wynik polecenia 'ls -l' do pliku $ls - l > lista_plikow.txt$

2.6.1 Wyszukiwanie tekstu: polecenie `grep`

Służy do wyszukiwania w tekście linii pasujących do zadanego wzorca. Znajdź linie zawierające "error"w pliku log.txt $grep"error"log.txt$

Ignoruj wielkość liter (-i) $grep - i"Error"log.txt$

Pokaż numery linii (-n) $grep - n"error"log.txt$

Policz, ile jest pasujących linii (-c) $grep - c"error"log.txt$

Pokaż linie, które NIE zawierają wzorca (-v, invert match) $grep - v"info"log.txt$

Szukaj rekurencyjnie we wszystkich plikach w katalogu (-r) $grep - r"TODO"/sciezka/do/projektu/$

2.6.2 Wyszukiwanie plików – Polecenie `find`

W pracy z dużymi projektami lub na serwerach często musimy znaleźć pliki o określonej nazwie, typie czy rozmiarze. Polecenie `find` jest do tego idealnym narzędziem. Znajdź wszystkie pliki z rozszerzeniem .txt w bieżącym katalogu i podkatalogach $find. - name" * .txt"$

Znajdź wszystkie katalogi (-type d) o nazwie "Documents"w całym systemie (/) $find/ - typed - name"Documents"$

Znajdź pliki większe niż 100MB w katalogu domowym ( ) $find\ -size + 100M$

[a4paper, 12pt]article

[utf8]inputenc [T1]fontenc [polish]babel geometry xcolor hyperref graphicx fancyhdr beramono tabularx

minted [most]tcolorbox minted,breakable

a4paper, left=2.5cm, right=2.5cm, top=2.5cm, bottom=2.5cm titlebgHTML4A90E2

colorlinks=true, linkcolor=titlebg, urlcolor=titlebg, citecolor=titlebg,

[L]Wprowadzenie do pracy w systemie Linux [C]12

definitionbox[1] colback=black!5, colframe=titlebg, fonttitle=**, coltitle=white, colbacktitle=titlebg, title=1, boxrule=1pt, arc=4mm, bottomrule=0.5pt, toprule=0.5pt, breakable**

importantbox[1] colback=yellow!10, colframe=orange!80!black, fonttitle=**, title=1, boxrule=1pt, arc=2mm, breakable**

terminal listing only, listing engine=minted, minted language=bash, minted style=vs, minted options= autogobble, breaklines, fontsize=, colback=black!70, colframe=black!50, breakable, arc=3mm, boxrule=1pt, left=5mm, top=3mm, bottom=3mm, title=[on line, size=small, boxrule=0pt, colback=red!80!black, arc=3mm] [on line, size=small, boxrule=0pt, colback=yellow!80!black, arc=3mm] [on line, size=small, boxrule=0pt, colback=green!70!black, arc=3mm] **Terminal – Bash** ,

Wprowadzenie do pracy w środowisku Linux

Część 2: Automatyzacja i Zarządzanie Systemem

Opracowanie dla studentów matematyki 2 stycznia 2026

Spis treści

3 Skrypty Bash – Wprowadzenie do automatyzacji

Skrypt powłoki (shell script) Jest to plik tekstowy zawierający sekwencję poleceń, które są wykonywane przez powłokę linia po linii. Skrypty pozwalają zautomatyzować złożone lub powtarzalne zadania, takie jak przygotowywanie danych do analizy, uruchamianie symulacji numerycznych, tworzenie kopii zapasowych czy generowanie cyklicznych raportów.

Zamiast ręcznie wpisywać dziesięciu poleceń, aby pobrać dane, przetworzyć je i wygenerować wykres, możemy zapisać je w skrypcie i uruchomić za pomocą jednej komendy.

3.1 Tworzenie i edycja skryptu: edytor `nano`

Do tworzenia i edycji plików tekstowych w terminalu służą edytory tekstu. Jednym z najprostszych jest `nano`. Otwórz (lub utwórz) plik o nazwie 'analiza.sh' w edytorze nano *nanoanaliza.sh* Po otwarciu edytora, na dole ekranu widoczne są najważniejsze skróty klawiszowe (znak ^ oznacza klawisz `Ctrl`).

- `^O` (Ctrl+O): Zapisz plik (*Write Out*).
- `^X` (Ctrl+X): Wyjdź z edytora (*Exit*).

Wpiszmy w edytorze `nano` treść naszego pierwszego skryptu: [frame=lines,framesep=2mm,fontsize=]b
!/bin/bash Prosty skrypt analityczny 1. Tworzy katalog na wyniki 2. Zapisuje w nim log z datą rozpoczęcia 3. Symuluje długotrwałe obliczenia

echo "Rozpoczynam analizę..."mkdir -p wyniki$_a$*nalizydate* > *wyniki*$_a$*nalizy/log.txtecho*"*Przeprowadzamsymulacj*..
wyniki$_a$*nalizy/log.txtdate* >> *wyniki*$_a$*nalizy/log.txt*

4 System uprawnień plików

Zanim uruchomimy nasz skrypt, musimy nadać mu prawo do wykonania. W Linuksie każdy plik i katalog ma precyzyjnie określone uprawnienia, które decydują o tym, kto i w jaki sposób może z nim wchodzić w interakcję.

Podstawą systemu uprawnień są trzy fundamentalne prawa, reprezentowane przez litery:

- **r (read)** – Prawo do **odczytu**.
  - Dla pliku: pozwala na przeglądanie jego zawartości (np. poleceniem `cat`).
  - Dla katalogu: pozwala na wylistowanie jego zawartości (np. poleceniem `ls`).
- **w (write)** – Prawo do **zapisu**.
  - Dla pliku: pozwala na modyfikowanie jego zawartości (edycję, nadpisywanie).

  - Dla katalogu: pozwala na tworzenie, usuwanie i zmianę nazw plików wewnątrz tego katalogu.

- **x (execute)** – Prawo do **wykonania**.
  - Dla pliku: pozwala na uruchomienie go jako programu lub skryptu.
  - Dla katalogu: pozwala na "wejście"do niego (np. poleceniem `cd`).

4.1 Odczyt uprawnień: Anatomia `ls -l`

Wynik polecenia `ls -l` zawiera szczegółowe informacje, w tym 10-znakowy ciąg opisujący uprawnienia: $ls - lanaliza.sh - rw - r - - r - - 1 student users 215 pa 2610 : 30 analiza.sh$

Analiza uprawnień: `-rw-r-r-` Ten 10-znakowy ciąg dzielimy na cztery części:

Znak 1	**Znaki 2-4**	**Znaki 5-7**	**Znaki 8-10**
`-`	`rw-`	`r-`	`r-`
Typ pliku	Właściciel	Grupa	Inni

- **Typ pliku**: `-` oznacza zwykły plik, `d` to katalog (directory).
- **Właściciel (user)**: Uprawnienia dla właściciela pliku. Tutaj: odczyt (`r`) i zapis (`w`).
- **Grupa (group)**: Uprawnienia dla grupy. Tutaj: tylko odczyt (`r`).
- **Inni (others)**: Uprawnienia dla pozostałych. Tutaj: tylko odczyt (`r`).
- Myślnik `-` w miejscu uprawnienia oznacza jego brak.

4.2 Zmiana uprawnień: polecenie `chmod`

Polecenie `chmod` (change mode) pozwala modyfikować te uprawnienia.

Zobaczmy obecne uprawnienia $ls - lanaliza.sh - rw - r - - r - - 1 student users 215 pa 2610 : 30 analiza.sh$

Nadajmy właścicielowi prawo do wykonania. $chmod rwxr - xr - x analiza.sh$

Sprawdźmy ponownie. Znak ‘x‘ został dodany, a nazwa pliku zmieniła kolor. $ls - lanaliza.sh - rwxr - xr - x 1 student users 215 pa 2610 : 32 analiza.sh$

Uprawnienia jako system ósemkowy Każde z trzech uprawnień (`r`, `w`, `x`) można przedstawić jako bit w liczbie 3-bitowej. Jest to naturalna konsekwencja potęg dwójki:

- **r** (read) = $2^2 = \mathbf{4}$
- **w** (write) = $2^1 = \mathbf{2}$
- **x** (execute) = $2^0 = \mathbf{1}$

Sumując te wartości, otrzymujemy jedną cyfrę (od 0 do 7) dla każdej kategorii użytkowników (właściciel, grupa, inni). Przykładowo, `chmod 754 plik.txt` oznacza:

- **7** dla właściciela: 4+2+1 → `rwx`
- **5** dla grupy: 4+0+1 → `r-x`
- **4** dla innych: 4+0+0 → `r--`

Zobaczmy obecne uprawnienia $ls-lanaliza.sh-rw-r--r--1studentusers215pa2610:30analiza.sh$

Nadajmy właścicielowi prawo do wykonania. Chcemy: rwx r-x r-x -> (4+2+1)(4+0+1)(4+0+1) -> 755 $chmod755analiza.sh$

Sprawdźmy ponownie. Znak 'x' został dodany, a nazwa pliku zmieniła kolor. $ls-lanaliza.sh-rwxr-xr-x1studentusers215pa2610:32analiza.sh$

[colback=black!5, colframe=titlebg, title=Pełna tabela systemu ósemkowego uprawnień (0-7), halign=center] |c|c|c|X|

Liczba (Octal)	Postać binarna (rwx)	Suma	Znaczenie uprawnień
0	`--`	0+0+0	Brak jakichkolwiek uprawnień. Plik jest całkowicie niedostępny.
1	`--x`	0+0+1	Tylko wykonanie. Umożliwia wejście do katalogu lub uruchomienie pliku binarnego, ale nie pozwala na odczyt jego zawartości.
2	`-w-`	0+2+0	Tylko zapis. Rzadko używane samodzielnie, ponieważ aby zapisać plik, zazwyczaj trzeba go najpierw odczytać.
3	`-wx`	0+2+1	Zapis i wykonanie. Pozwala na modyfikację i uruchomienie pliku.
4	`r--`	4+0+0	Tylko odczyt. Typowe uprawnienie dla plików z danymi, które nie powinny być modyfikowane przez wszystkich.
5	`r-x`	4+0+1	Odczyt i wykonanie. Standardowe uprawnienie dla programów i katalogów, do których potrzebny jest dostęp.
6	`rw-`	4+2+0	Odczyt i zapis. Typowe uprawnienie dla plików, nad którymi pracujemy (np. dokumenty, kod źródłowy).
7	`rwx`	4+2+1	Pełne uprawnienia. Zapewnia pełną kontrolę nad plikiem lub katalogiem.

4.3 Uruchamianie skryptu

Gdy plik ma już prawo do wykonania, możemy go uruchomić, podając jego ścieżkę. Kropka (.) to skrót oznaczający bieżący katalog $./analiza.shWynik : Rozpoczynamanaliz...Przeprowa$

5 Monitorowanie i zarządzanie procesami

Proces To uruchomiona instancja programu. Każdy program, który działa w systemie (nawet sam terminal), ma co najmniej jeden proces. Każdy proces ma unikalny, numeryczny identyfikator (**PID** - Process ID) oraz określony stan.

5.1 Stany procesów

- **Running (R)**: Proces jest aktualnie wykonywany przez procesor lub czeka w kolejce na swoją turę.
- **Sleeping (S)**: Proces czeka na zdarzenie (np. dane z dysku, odpowiedź z sieci). Większość procesów przez większość czasu jest w tym stanie.
- **Stopped (T)**: Proces został zatrzymany (np. przez użytkownika) i może być wznowiony.
- **Zombie (Z)**: Proces zakończył działanie, ale jego wpis w tablicy procesów wciąż istnieje, ponieważ proces nadrzędny nie odczytał jego statusu zakończenia.

5.2 Interaktywny monitoring: polecenie `top`

`top` to fundamentalne narzędzie diagnostyczne. Poza wyświetlaniem listy procesów, pozwala na interaktywne zarządzanie widokiem. Interaktywne polecenia w `top` Będąc w `top`, wciśnij klawisz, aby zmienić zachowanie:

- `M` (duże M): Sortuj procesy według użycia pamięci (%MEM).
- `P` (duże P): Sortuj procesy według użycia CPU (%CPU) – domyślne.
- `k`: "Zabij"proces. `top` zapyta o PID procesu do zabicia.
- `h`: Wyświetl pomoc.
- `q`: Wyjdź.

Ciekawostka: `htop` `htop` to nowocześniejsza, bardziej kolorowa i interaktywna wersja `top`. Oferuje m.in. łatwiejsze przewijanie i zabijanie procesów za pomocą klawiszy funkcyjnych.

5.3 Wyszukiwanie i zabijanie procesów

Ręczne szukanie PID w `top` jest nieefektywne. Lepiej użyć dedykowanych narzędzi.

5.3.1 Wyszukiwanie procesów: `pgrep`

Polecenie `pgrep` (process grep) wyszukuje procesy po nazwie i zwraca ich PID. Uruchommy w tle proces, który będzie długo działał *sleep*600[1]12345

Znajdźmy PID procesu o nazwie ‘sleep‘ *pgrepsleep*12345

5.3.2 Zabijanie procesów: sygnały i `kill`

Polecenie `kill` nie "zabija"procesu wprost. Wysyła do niego **sygnał**, czyli komunikat systemowy. Proces może na sygnał zareagować, np. grzecznie się zamykając.

[colback=black!5, colframe=titlebg, width=0.95] |c|c|X|

Sygnał	Numer	Opis i zastosowanie
`SIGTERM`	15	**Terminate (zakończ)**. Domyślny, "uprzejmy"sygnał. Prosi proces o zakończenie pracy, dając mu szansę na zapisanie danych i posprzątanie. To pierwsza próba zamknięcia programu.
`SIGKILL`	9	**Kill (zabij)**. Sygnał ostateczny, "brutalny". Nie może być zignorowany. Jądro systemu natychmiast usuwa proces z pamięci. Używany, gdy proces się zawiesił i nie reaguje na `SIGTERM`.
`SIGINT`	2	**Interrupt (przerwij)**. Sygnał wysyłany po naciśnięciu `Ctrl+C` w terminalu.

Znajdź PID procesu ‘sleep‘ *pgrepsleep*12345

Wyślij domyślny sygnał SIGTERM (prośba o zamknięcie) *kill*12345

Jeśli proces nie reaguje, użyj SIGKILL kill -9 <PID> LUB kill -SIGKILL <PID> *kill* − 912345

Polecenie `pkill` `pkill` łączy w sobie funkcjonalność `pgrep` i `kill`. Znajduje procesy po nazwie i od razu wysyła do nich sygnał. Jest bardzo wygodne, ale też bardziej ryzykowne – można przypadkowo zabić wiele procesów naraz.

```
pkill -9 nazwa_procesu
```