

Wprowadzenie do pracy w środowisku Linux

Notatki z zajęć laboratoryjnych

CZEŚĆ 1

Spis treści

Wprowadzenie	3
1 Podstawy teoretyczne: System operacyjny	3
1.1 Czym jest system operacyjny?	3
1.2 Serce systemu, czyli jądro (kernel)	3
1.3 Dwa światy: Linux vs. Windows	3
1.3.1 Porównanie jąder systemowych	3
1.3.2 Zalety i wady obu systemów	4
1.3.3 Przykłady wykorzystania	4
1.4 Świat Linuksa: Dystrybucje	4
2 Podstawowe komendy w systemie Linux	4
2.1 Powłoka systemowa: Bash	4
2.2 Struktura poleceń: opcje i argumenty	5
2.3 Polecenia informacyjne	5
2.4 Nawigacja i listowanie plików	6
2.5 Zarządzanie plikami i katalogami	7
2.6 Przeglądanie i przetwarzanie plików	9
2.6.1 Wyszukiwanie tekstu: polecenie grep	10
2.6.2 Wyszukiwanie plików – Polecenie find	10

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 **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 nadzędne 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 mikrojądro (dla stabilności). Kod jądra Windows jest zamknięty i stanowi własność firmy Microsoft.

1.3.2 Zalety i wady obu systemów

Windows	Linux
<p>Zalety:</p> <ul style="list-style-type: none">• Prostota obsługi i intuicyjność.• Ogromna kompatybilność z oprogramowaniem i grami. <p>Wady:</p> <ul style="list-style-type: none">• System jest płatny (licencja).• Postrzegany jako bardziej podatny na wirusy.	<p>Zalety:</p> <ul style="list-style-type: none">• Jest darmowy i otwartoźródłowy.• Wysoka stabilność i bezpieczeństwo. <p>Wady:</p> <ul style="list-style-type: none">• 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.

- **Debian**: Znany ze swojej stabilności, często używany na serwerach.
- **Ubuntu**: Bardzo popularny, uważany za przyjazny dla początkujących.
- **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 góre/dół), autouzupełnianie (klawisz **Tab**) oraz możliwość tworzenia skryptów automatyzujących zadania.

2.2 Struktura poleceń: opcje i argumenty

Praca w terminalu polega na wpisywaniu polecen 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:

- **Krótką formą**: jeden myślnik i jedna litera, np. **-l**. Krótkie opcje można łączyć. Zamiast pisać **ls -l -h -a**, można to skrócić do **ls -lha**.
- **Długą formą**: 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).



Terminal – Bash

```
# Wyświetla podany tekst na ekranie  
$ echo 'Hello World'
```

Polecenie **whoami**

Wyświetla nazwę aktualnie zalogowanego użytkownika.



Terminal – Bash

```
# Wyświetla nazwę zalogowanego użytkownika  
$ whoami
```

Polecenie **groups**

Pokazuje nazwy grup, do których należy bieżący użytkownik.



Terminal – Bash

```
# Pokazuje grupy, do których należy użytkownik  
$ groups
```

Polecenie **date**

Wyświetla aktualną datę i godzinę. Jego format można kontrolować.



Terminal – Bash

```
# Wyświetla bieżącą datę i godzinę  
$ date  
  
# Wyświetla datę w formacie ROK-MIESIĄC-DZIEŃ  
$ date +"%Y-%m-%d"
```

Polecenie **uname**

Wyświetla informacje o systemie operacyjnym i jego jądrze.



Terminal – Bash

```
# 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).

```
Terminal – Bash
```

```
# Sprawdź, gdzie jesteś  
$ pwd
```

Polecenie **cd**

Zmienia bieżący katalog (change directory).

```
Terminal – Bash
```

```
# Zmień katalog na podany  
$ cd /sciezka/do/katalogu  
  
# Przejdz do katalogu domowego użytkownika  
$ cd ~  
  
# Wróć do poprzedniego katalogu  
$ cd -
```

Polecenie **ls**

Wyświetla listę plików i katalogów w bieżącej lokalizacji.

```
Terminal – Bash
```

```
# 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.

```
Terminal – Bash
```

```
# Utwórz pusty plik  
$ touch nowy_plik.txt
```

Polecenie **mkdir**

Tworzy nowy katalog (make directory).

```
Terminal – Bash

# Utwórz nowy katalog
$ mkdir nowy_katalog

# Utwórz całą ścieżkę katalogów (parents)
$ mkdir -p A/B/C
```

Polecenie **cp**

Kopiuje pliki lub katalogi.

```
Terminal – Bash

# Kopiuj plik (cp <źródło> <cel>)
# -v: tryb gadatliwy (verbose), pokazuje co robi
# -i: tryb interaktywny, pyta przed nadpisaniem
$ cp -iv plik.txt /tmp/kopia_pliku.txt

# Kopiuj cały katalog (opcja -r, recursive)
$ cp -r moj_folder/ /tmp/
```

Polecenie **mv**

Przenosi lub zmienia nazwę plików i katalogów (move).

```
Terminal – Bash

# Zmień nazwę pliku (mv <stara> <nowa>)
$ mv plik.txt dokument.txt

# Przenieś plik do innego katalogu z trybem gadatliwym
$ mv -v dokument.txt /tmp/
```

Polecenie **rm**

Usuwa pliki lub katalogi (remove).

```
Terminal – Bash

# Usuń plik
$ rm plik.txt

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

```
Terminal – Bash  
# Wyświetl całą zawartość pliku z numerami linii (-n)  
$ cat -n skrypt.sh
```

Polecenie **head**

Wyświetla pierwsze linie pliku (domyślnie 10).

```
Terminal – Bash  
# Wyświetl pierwsze 5 linii pliku  
$ head -n 5 /var/log/syslog
```

Polecenie **tail**

Wyświetla ostatnie linie pliku (domyślnie 10).

```
Terminal – Bash  
# Wyświetl ostatnie 3 linie pliku  
$ tail -n 3 /var/log/syslog  
  
# Śledź plik na żywo (idealne do logów, -f to follow)  
$ tail -f /var/log/syslog
```

Polecenie **wget**

Pobiera pliki z internetu.

```
Terminal – Bash  
# Pobierz plik i zapisz pod inną nazwą (-O)  
$ wget -O pan_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.

```
Terminal – Bash

# 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.

```
Terminal – Bash

# 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.

```
Terminal – Bash

# Znajdź wszystkie pliki z rozszerzeniem .txt w bieżącym katalogu i
# w podkatalogach
$ find . -name "*.txt"

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

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