Technologie informacyjne i komunikacyjne

Wykład 2, dn. 2.03.2026

Źródło obrazka: pngtree

Informacje techniczne

Adrianna Tartas
Zakład Fizyki Biomedycznej
pok. 4.59

atartas@fuw.edu.pl

fuw.edu.pl/~atartas/TIK.html

UAKTUALNIENIE

cd. Linuxa

W systemie Linux **zwykły użytkownik nie ma dostępu** do wszystkich plików i ustawień systemowych. To zabezpieczenie chroni system przed:

- przypadkowym usunięciem ważnych plików
- instalacją niebezpiecznego oprogramowania
- zmianą konfiguracji systemu

Aby wykonać **zadanie administracyjne**, używa się `sudo`.

Jak działa `sudo`?

Po wpisaniu komendy z `sudo`:

```
sudo apt update
```

System:

- Sprawdza, czy użytkownik ma uprawnienia do używania `sudo`.
- Prosi o hasło użytkownika.
- Wykonuje polecenie z uprawnieniami administratora.

Przykłady użycia `sudo`?

Aktualizacja systemu:

```
sudo apt upgrade
```

Instalacja programu:

```
sudo apt install firefox
```

Edycja pliku systemowego:

```
sudo nano /etc/hosts
```

Kim jest „root”?

W Linuxie istnieje **specjalne konto administratora** o nazwie **root**.

Ma ono **pełną kontrolę** nad systemem.

W wielu dystrybucjach (np. Ubuntu) konto root jest domyślnie zablokowane, a administracja odbywa się właśnie przez `sudo`.

Zalety `sudo`:

- **Zwiększa bezpieczeństwo** systemu
- Rejestruje wykonane polecenia
- Ogranicza dostęp administracyjny tylko do wybranych użytkowników
- Zmniejsza ryzyko uszkodzenia systemu

Wady / ryzyko:

- Nieostrożne użycie może uszkodzić system
- Daje pełne uprawnienia administratora
- Błędna konfiguracja może narazić system na ataki

Ciekawostki:

- `sudo` powstało w 1980 roku
- Jest **standardem** w większości dystrybucji Linuxa
- Używane także w systemach opartych na Unix

Podsumowanie

`sudo` to narzędzie umożliwiające **bezpieczne wykonywanie poleceń administracyjnych** w systemie Linux.

Pozwala zachować równowagę między **bezpieczeństwem a wygodą** zarządzania systemem.

Typy danych

- Wartości logiczne (prawda/fałsz)
- Znaki (a-z, A-Z, 0-9,:,;@,#,$,%,...)
- Liczby (całkowite i niecałkowite)
- Obrazy
- Sygnały dźwiękowe

Reprezentacja danych

BIT = **bi**nary digi**t** (0 lub 1)

- Wszystkie dane są zapisywane przy pomocy bitów
- Komputer operuje wyłącznie na grupach bitów tworzących liczby binarne
- 8 bitów to bajt
- 1 bajt reprezentuje 1 znak

Kodowanie znaków

- Każdy znak jest reprezentowany przez liczbę stanowiącą jego numer w tablicy kodowej
- Tablice kodowe
  - ASCII: 128 pozycji, w tym małe i wielkie litery alfabetu łacińskiego, cyfry, znaki przestankowe
  - Rozszerzone ASCII: 256 pozycji, w tym pierwsze 128 takie jak w ASCII, a następne 128 zawiera znaki narodowe (np. ą, ę, ó, ś, ...) lub inne symbole
  - UNICODE: pierwotnie 216 = 65 536, a obecnie 232 = 4 294 967 296 pozycji reprezentujących wszystkie znaki używane na świecie

Jednostki informacji

bit	b	0 lub 1
bajt	B	od 00000000 do 11111111
		(2^8 kombinacji czyli 256 znaków)

Przykład zapisu liczby

Jeśli zapiszemy liczbę $13_{\mathbf{10}}$ jako bajt (8 bitów), musimy uzupełnić zapis zerami z lewej strony:

$13_{\mathbf{10}}$ jako bajt:

$$00001101_{\mathbf{2}}$$

Wyjaśnienie:

$$00001101_{\mathbf{2}} = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^{\mathbf{0}} = 13_{\mathbf{10}}$$

Jednostki informacji

bit	b	0 lub 1
bajt	B	od 00000000 do 11111111 (2^8 kombinacji czyli 256 znaków)
kilobajt	kB	2^{10} bajtów czyli 1024 B
megabajt	MB	2^{10} kilobajtów czyli 1024 kB
gigabajt	GB	2^{10} megabajtów czyli 1024 MB
terabajt	TB	2^{10} gigabajtów czyli 1024 GB

Gdzie są moje gigabajty?

Producent:

250 GB

system dziesiętny

1kB = 1000 bajtów

$$\frac{250\ 000\ 000\ 000 \text{ bajtów}}{1000*1000*1000}$$

Komputer:

232,83 GB

system binarny

1kB = 1024 bajty

$$\frac{250\ 000\ 000\ 000 \text{ bajtów}}{1024*1024*1024}$$

Zakres liczb binarnych

Największa możliwa liczba to taka, w której wszystkie bity mają wartość 1, czyli każda pozycja (każda waga 2^0, 2^1, ..., $2^{n-1)}$ bierze udział w tworzeniu liczby.

Ma ona postać:

$$111...111_{\mathbf{2}} \text{ (n jedynek)}$$

Jej wartość dziesiętna wynosi:

$$2^n - 1$$

Przykład: Dla 8 bitów

$$11111111_{\mathbf{2}} = 255_{\mathbf{10}}$$

czyli maksymalna liczba zapisana na 8 bitach to $2^{\mathbf{8}} - 1 = 255$.

Zakres liczb binarnych

Jaką największą liczbę dziesiętną można przedstawić za pomocą 34 bitów?

Zakres liczb binarnych

Jaką największą liczbę dziesiętną można przedstawić za pomocą 34 bitów?

$$2^{34} - 1 = 17179869184 - 1 = 17179869183$$

Dodawanie w systemie binarnym

Dodawanie binarne działa podobnie jak w systemie dziesiętnym, ale używa tylko cyfr 0 i 1.

Podstawowe reguły:

```
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10          (0 i przeniesienie 1)
1 + 1 + 1 = 11      (1 i przeniesienie 1)
```

Dodawanie w systemie binarnym

Dodajmy:

$1011_{\mathbf{2}}$ $(11_{\mathbf{10}})$

\+ $110_{\mathbf{2}}$ $(6_{\mathbf{10}})$

Wyrównujemy do 4 bitów:

$$\begin{array}{rcccc} & 1 & 0 & 1 & 1 \\ + & 0 & 1 & 1 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 \end{array}$$

Wynik:

$10001_{\mathbf{2}} = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 17_{\mathbf{10}}$

Odejmowanie w systemie binarnym

Metoda „pisemna” (z pożyczką): Mechanizm jest taki sam jak w systemie dziesiętnym – w binarnym jest po prostu prostszy (tylko cyfry 0 i 1).

```
0 – 0 = 0
1 – 0 = 1
1 – 1  = 0
0 – 1 = 1
```

+ pożyczka

Gdy odejmujemy 0 – 1, „pożyczamy” 1 z następnej pozycji. Pożyczka oznacza, że w następnej kolumnie musimy odjąć dodatkowe 1.

Odejmowanie w systemie binarnym

Przykład odejmowania binarnego (z pożyczką):

	1	0	0	0	$(8_{\mathbf{10}})$
-	0	0	0	1	$(1_{\mathbf{10}})$
	0	1	1	1	$(7_{\mathbf{10}})$

Co się dzieje?

- 0 – 1 → nie można → pożyczamy z lewej strony
- $10_{\mathbf{2}}$ – 1 = 1
- Pożyczka „przechodzi” przez kolejne zera
- Ostateczny wynik: $0111_{\mathbf{2}}$

Niedomiar

Jeśli od liczby mniejszej odejmiemy większą, otrzymamy wynik ujemny. W naturalnym systemie binarnym nie ma jednak możliwości zapisu liczb ujemnych. Sprawdźmy więc, co się stanie, gdy od liczby 0 odejmiemy 1 i ograniczymy wynik do 8 bitów.

$$\begin{array}{rcccccccc} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}$$

Otrzymujemy same jedynki, a pożyczka nigdy nie znika. Taka sytuacja nazywa się **niedomiarem (ang. underflow)** i występuje zawsze, gdy wynik operacji arytmetycznej jest mniejszy od dolnego zakresu reprezentacji liczb binarnych – w przypadku naturalnego kodu dwójkowego oznacza to wynik mniejszy od zera.

Kodowanie liczb ze znakiem

Naturalny system binarny (NBS):

- Pozwala na zapis tylko liczb dodatnich i zera.
- Każda liczba reprezentowana jest jako ciąg bitów 0 i 1.

Problem liczb ujemnych:

- W NBS nie da się zapisać wartości ujemnych.
- Aby reprezentować liczby ze znakiem, trzeba dodać informację o znaku do bitów liczby.
- Główne cele kodowania ze znakiem:
  - Możliwość zapisu liczb ujemnych i dodatnich.
  - Ułatwienie wykonywania operacji arytmetycznych na komputerze.

Kod znak-moduł

Najstarsza metoda.

Najstarszy bit (MSB) – bit znaku:

0 → liczba dodatnia

1 → liczba ujemna

Pozostałe bity – wartość liczby (moduł).

Przykład (8 bitów):

5 → 0 0 0 0 0 1 0 1

-5 → 1 0 0 0 0 1 0 1

Kod uzupełnień do jedności (U1)

Ujemną liczbę tworzymy przez odwrócenie wszystkich bitów liczby dodatniej.

Operacje arytmetyczne łatwiejsze niż w kodzie znak-moduł, ale nadal mają problemy przy dodawaniu 0.

Przykład (8 bitów):

5 → 0 0 0 0 1 0 1

-5 → 1 1 1 1 0 1 0

Kod uzupełnień do dwóch (U2)

- Najczęściej stosowana metod
- Najpierw odwracamy bity liczby dodatniej, potem dodajemy 1.
- Pozwala bezpośrednio dodawać i odejmować liczby dodatnie i ujemne.
- Współczesne komputery używają wyłącznie tego kodu.

Przykład (8 bitów):

$$5 \rightarrow 0\ 0\ 0\ 0\ 1\ 0\ 1$$

$$-5 \rightarrow \begin{matrix} 1\ 1\ 1\ 1\ 0\ 1\ 0 \\ 1\ 1\ 1\ 1\ 0\ 1\ 1 \end{matrix}$$

Metoda	Zalety	Wady
Znak-moduł	Prosta i intuicyjna Łatwo odczytać znak i wartość liczby	Trudne operacje arytmetyczne (dodawanie/odejmowanie) Dwa zera: +0 i -0
Uzupełnienie do jedności (U1)	Operacje arytmetyczne łatwiejsze niż w znak-moduł	Nadal występują problemy przy dodawaniu liczb ujemnych i zer Dwa zera: +0 i -0
Uzupełnienie do dwóch (U2)	Najczęściej stosowane w komputerach Proste dodawanie i odejmowanie Tylko jedno zero	Dla początkujących może być mniej intuicyjne Minimalna komplikacja przy odczycie wartości ujemnej (trzeba odwrócić proces)

Reprezentacja liczb rzeczywistych - ułamki

$6.375_{(10)} = 6 \times 10^{0} + 3 \times 10^{-1} + 7 \times 10^{-2} + 5 \times 10^{-3}$ - system dziesiętny

$6.375_{(10)} = 101.01_{(2)}$

Konwersja do systemu binarnego

Liczba całkowita: $6 \rightarrow 0110_2$

Część ułamkowa: 0.375 (metoda mnożenia przez 2):

$$0.375 \times 2 = 0.75 \rightarrow 0$$
$$0.75 \times 2 = 1.5 \rightarrow 1$$
$$0.5 \times 2 = 1.0 \rightarrow 1$$

Czyli $0.375 \rightarrow 0.011_2$

Połączone: $6.375_{(10)} = 110.011_{(2)}$

Dziękuję za uwagę!