Technologie informacyjne i komunikacyjne

Wykład 1, dn. 23.02.2026



Źródło obrazka: pngtree

Informacje techniczne



Adrianna Tartas
Zakład Fizyki Biomedycznej
pok. 4.59



atartas@fuw.edu.pl



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

Zasady zaliczenia przedmiotu

Rozkład materiału na ćwiczeniach

2 tygodnie: Linux

3 tygodnie: LaTeX → **projekt** do domu: **10 pkt**

3 tygodnie: Mathematica → **sprawdzian** na zajęciach: **15 pkt**

30.04 grupy czwartkowe

4.05 grupa poniedziałkowa

4 tygodnie: Python → **sprawdzian** na zajęciach: **15 pkt**

8.06 grupa poniedziałkowa

11.06 grupy czwartkowe

Kolokwium **poprawkowe** w sesji - możliwość poprawy jednego działu.

Zasady zaliczenia przedmiotu

Punktacja końcowa

Projekty i sprawdziany: **40 pkt**

Egzamin (w formie testu) z wykładu (**18.06 ?**): **10 pkt**

Łącznie: **50 pkt**

Obecność na wykładzie (dopuszcza się 2 nieusprawiedliwione nieobecności) daje możliwość podejścia do egzaminu w **terminie zerowym** w dniu **8.06**.

Punkty	< 25	25–29	30–34	35–39	40–44	45–49	50
Ocena	2.0	3.0	3.5	4.0	4.5	5.0	5.0+

Zakres tematów:

1. Zapoznanie z systemem operacyjnym Linux i jego podstawową architekturą.

2. Praca w środowisku Linux z wykorzystaniem wiersza poleceń, w tym omówienie najważniejszych i najczęściej używanych poleceń.

3. Podstawy skryptów powłoki Bash, strumieni danych oraz przekierowań.

4. Wyrażenia regularne i ich praktyczne zastosowanie w narzędziach takich jak grep, sed oraz awk.

5. Podstawowe zagadnienia związane z dostępem do sieci i pracą w środowisku sieciowym.

6. Bezpieczeństwo i poufność w Internecie, w tym ochrona danych, prywatność użytkownika oraz podstawowe zagrożenia cyfrowe.

7. Wprowadzenie do systemu składu tekstu LaTeX oraz jego zastosowań w dokumentach naukowych.

8. Przygotowywanie prezentacji z wykorzystaniem LaTeX-a.

9. Podstawy pracy z oprogramowaniem Wolfram Mathematica.

10. Tworzenie wykresów i wizualizacja danych z wykorzystaniem biblioteki Matplotlib.

11. Wstęp do programowania w języku Python i jego zastosowań w analizie danych.

12. Wprowadzenie do zagadnień związanych ze Sztuczną Inteligencją, w tym podstawowe pojęcia, przykłady zastosowań oraz aspekty etyczne.

System operacyjny

Jest to podstawowe oprogramowanie, które zarządza działaniem komputera i umożliwia uruchamianie innych programów.

Można powiedzieć, że jest „pośrednikiem” między użytkownikiem i aplikacjami a sprzętem komputera.

Główne zadania systemu operacyjnego:

- Zarządzanie procesorem – przydziela czas procesora uruchomionym programom.
- Zarządzanie pamięcią – kontroluje wykorzystanie pamięci RAM.
- Zarządzanie plikami – organizuje dane na dysku w postaci plików i katalogów.
- Obsługa urządzeń – umożliwia korzystanie z klawiatury, myszy, drukarki, karty sieciowej itp.
- Zapewnienie bezpieczeństwa – kontroluje dostęp użytkowników do zasobów systemu.

Przykłady systemów operacyjnych

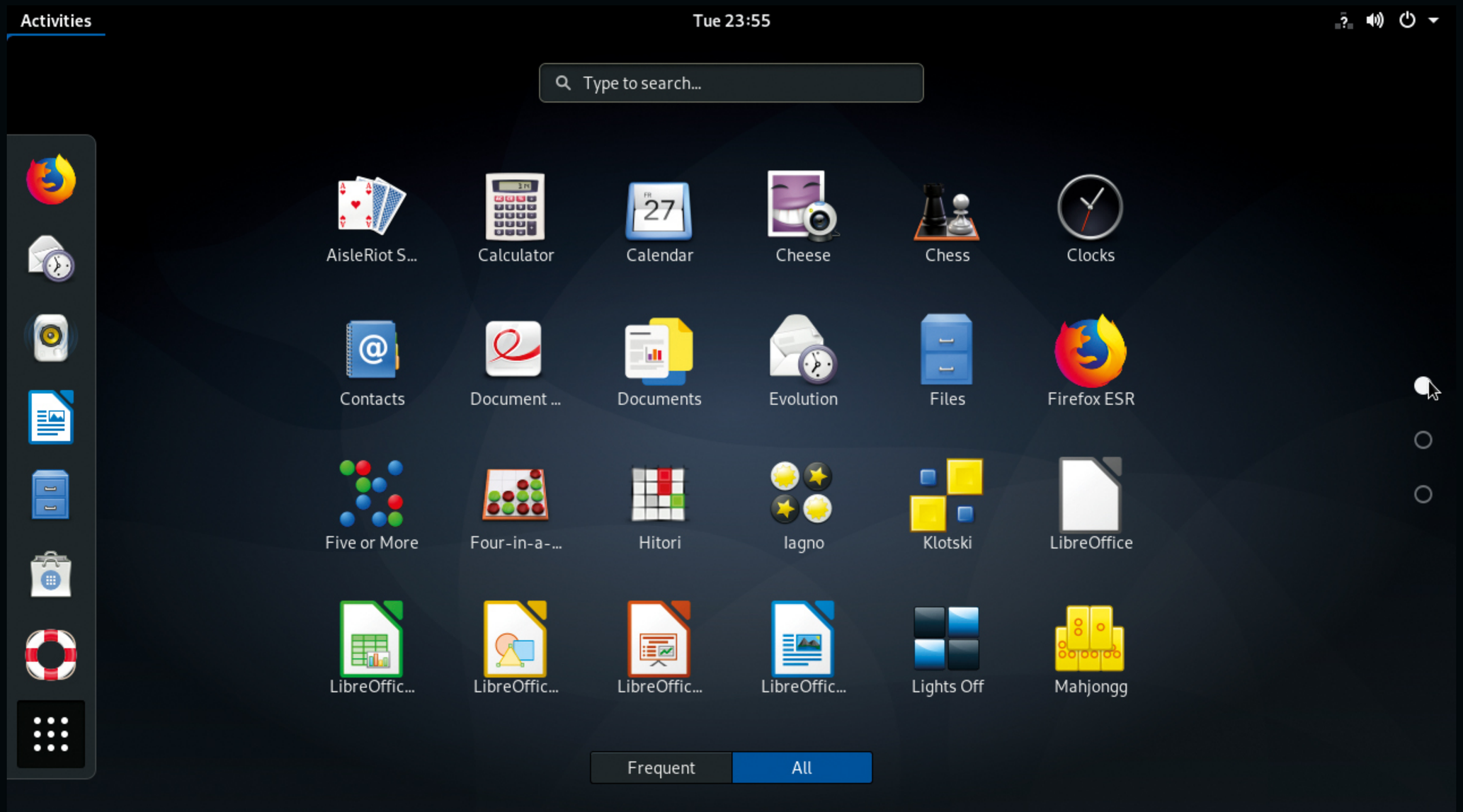
Microsoft Windows, macOS, **Linux**, Android

Linux to rodzina systemów operacyjnych typu Unix. Stanowi przykład wolnego i otwartego oprogramowania (FLOSS), a jego kod źródłowy może być swobodnie używany, modyfikowany oraz rozpowszechniany.

Przykłady dystrybucji:

Ubuntu, Debian, Fedora

Linux


Activities
Tue 23:55
Type to search...
AisleRiot S...
Calculator
Calendar
Cheese
Chess
Clocks
Contacts
Document ...
Documents
Evolution
Files
Firefox ESR
Five or More
Four-in-a-...
Hitori
Iagno
Klotski
LibreOffice
LibreOffic...
LibreOffic...
LibreOffic...
LibreOffic...
Lights Off
Mahjongg
Frequent
All

Historia Linuxa

Linus Torvalds to fiński student informatyki, który chciał stworzyć darmowy system operacyjny podobny do UNIX-a. Początkowo pisał go dla własnej nauki, ale w sierpniu 1991 roku ogłosił w internecie, że kod źródłowy jest dostępny dla każdego do testowania i rozwijania.

Linux od początku był **open source** – każdy mógł wprowadzać poprawki i dodawać funkcje. To przyciągnęło programistów z całego świata. Wkrótce powstały różne wersje Linuxa, zwane dystrybucjami, np. Debian, Red Hat, Slackware.

W latach 90. Linux zaczął być używany na serwerach, w superkomputerach i w urządzeniach wbudowanych. Dzięki **stabilności** i **bezpieczeństwu** stał się popularny w firmach i instytucjach. Pojawiły się też wersje dla zwykłych użytkowników, takie jak Ubuntu, które ułatwiły korzystanie z Linuxa na desktopie.

Dzisiaj Linux jest fundamentem wielu systemów – od serwerów i chmur, przez smartfony z Androidem, po superkomputery. Jest symbolem **wolnego oprogramowania**, współpracy globalnej i niezależności od dużych firm.

Filozofia Open Source

Open Source (otwarte oprogramowanie) to podejście do tworzenia i dystrybucji oprogramowania, w którym kod źródłowy jest dostępny dla każdego i może być: **przeglądany**, **modyfikowany**, **rozpowszechniany**.

Podstawowe zasady:

- **Wolny dostęp do kodu źródłowego**: Każdy może zobaczyć, jak program działa „od środka”.
- **Prawo do modyfikacji**: Użytkownicy mogą poprawiać błędy, dodawać funkcje, dopasowywać oprogramowanie do własnych potrzeb.
- **Prawo do dystrybucji**: Zmodyfikowane wersje mogą być rozpowszechniane, często również za darmo.
- **Współpraca i transparentność**: Społeczność może wspólnie rozwijać oprogramowanie, dzielić się doświadczeniem i poprawkami.

Filozofia Open Source

Różnica między FLOSS, Free Software i Open Source

FLOSS = Free/Libre Open Source Software - obejmuje zarówno ideę wolności (Free Software), jak i praktyczny dostęp do kodu (Open Source).

Free Software (FSF) – nacisk na wolność użytkownika („wolny” ≠ „darmowy”)

Open Source (OSI) – nacisk na dostęp do kodu i rozwój społecznościowy

Zalety open source

- Możliwość nauki i eksperymentowania
- Poprawa bezpieczeństwa przez wgląd w kod
- Wspólne rozwijanie projektów przez społeczność
- Brak uzależnienia od jednego dostawcy („vendor lock-in”)

System plików w Linuxie

System plików to sposób, w jaki Linux organizuje dane na dysku. Określa, jak pliki i katalogi są przechowywane, nazywane i zarządzane.

Linux używa hierarchicznego systemu katalogów, z głównym katalogiem / (root) na szczycie. Przykłady najważniejszych katalogów:

/bin – podstawowe programy systemowe

/home – katalogi domowe użytkowników

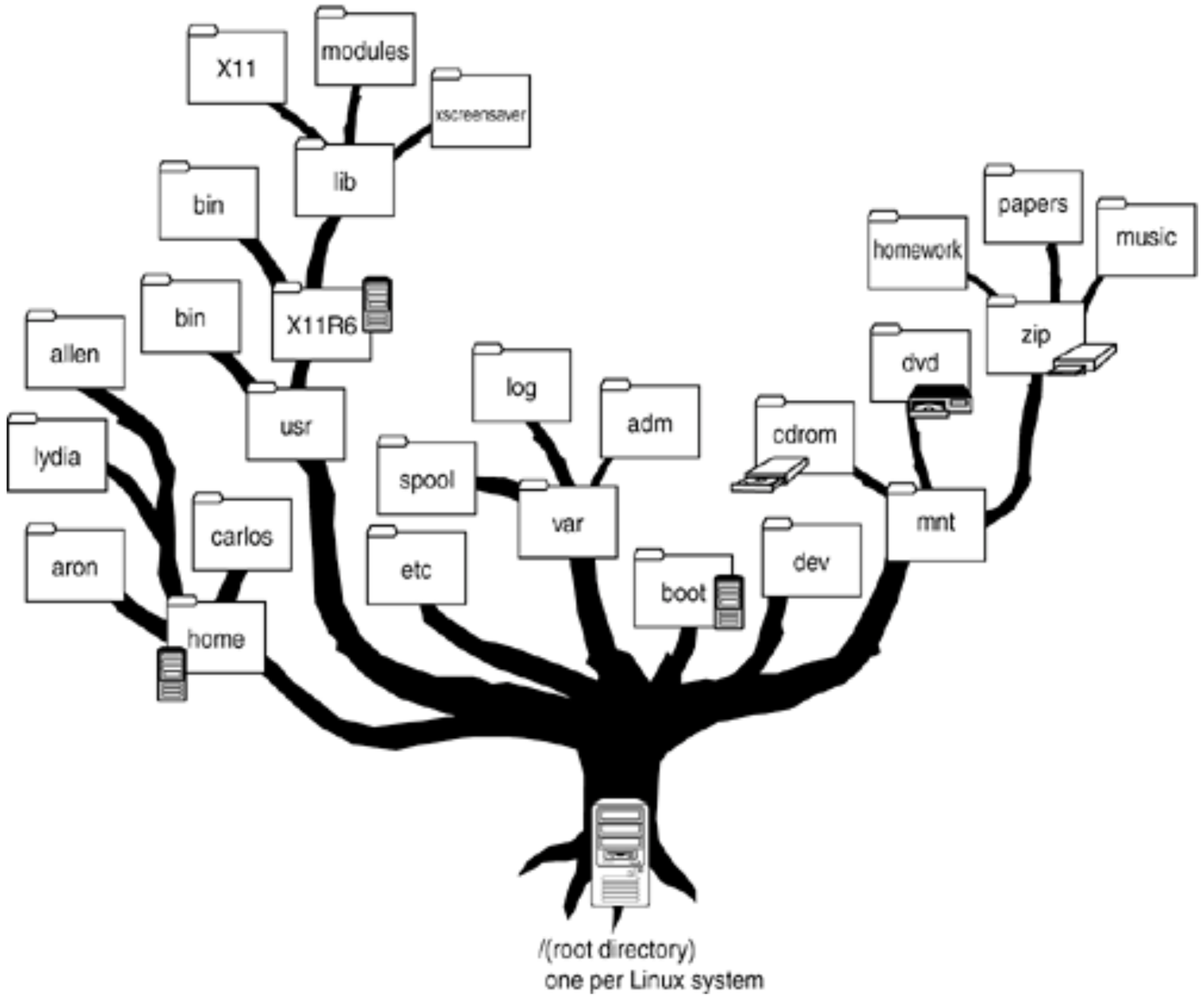
/etc – pliki konfiguracyjne systemu

/var – dane zmienne (logi, bazy danych)

/tmp – pliki tymczasowe

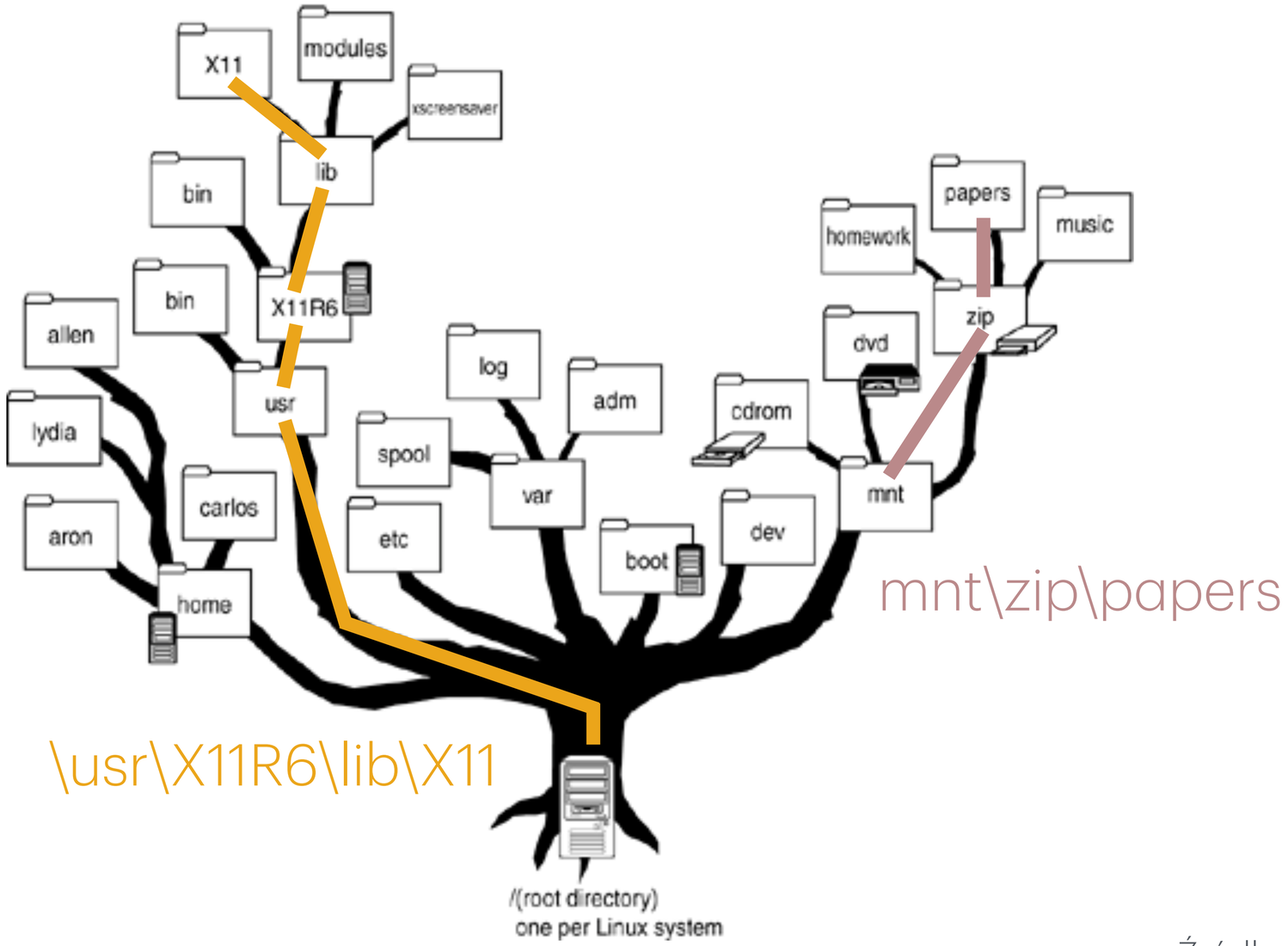
W Linuxie wszystko jest traktowane jak plik, nawet urządzenia (np. dyski, drukarki) są reprezentowane jako pliki w /dev.

System plików w Linuxie



Źródło obrazu: bit Warrior

System plików w Linuxie

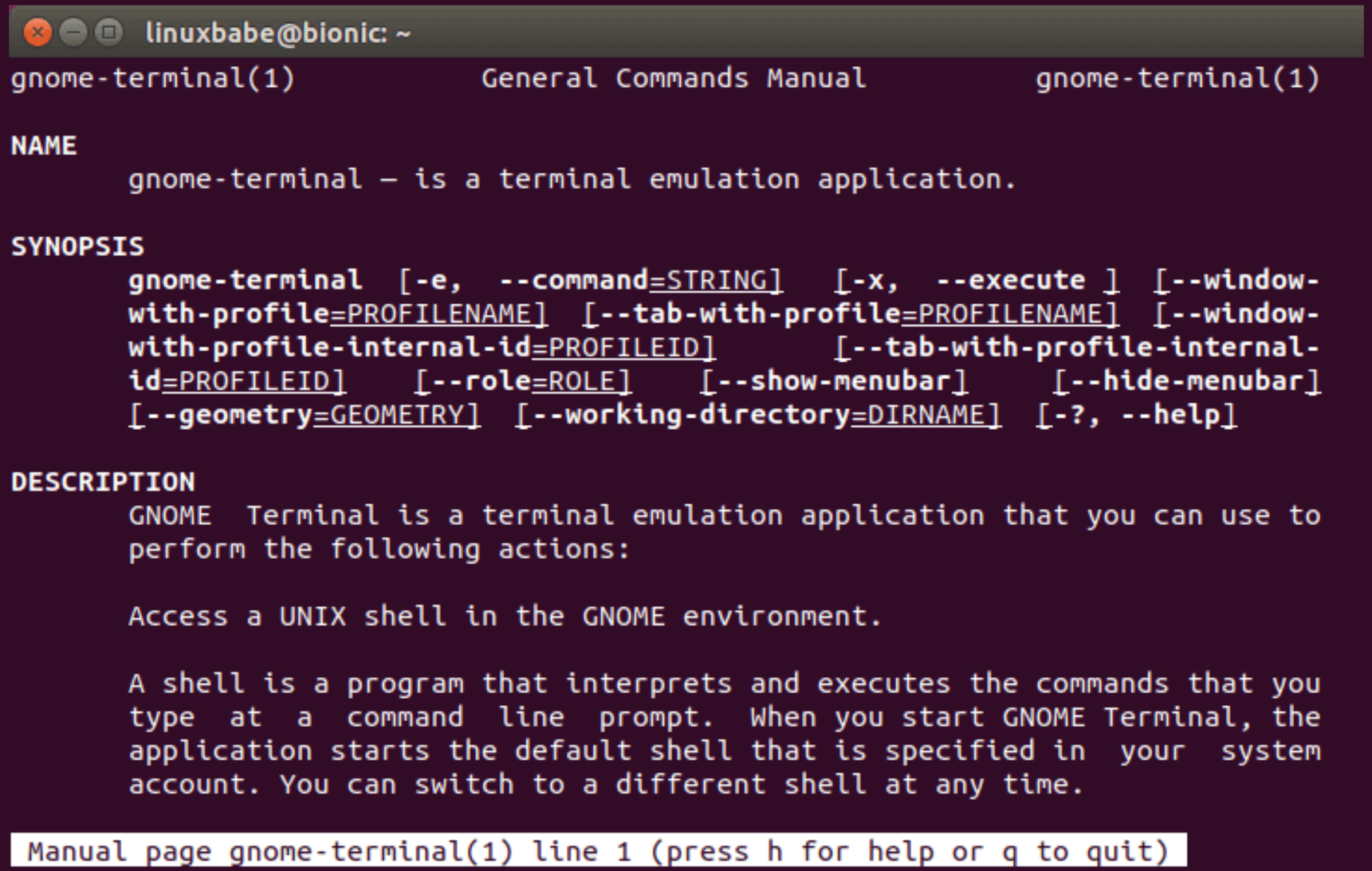


Źródło obrazu: bit Warrior

Terminal

...to okienko, w którym wpisujemy polecenia interpretowane przez system operacyjny, a dokładniej przez powłokę systemową (shell). Formalnie nazywa się je **emulatorem terminala**, ponieważ historycznie terminal był fizycznym urządzeniem – zazwyczaj monitorem i klawiaturą – służącym do komunikacji z komputerem w trybie tekstowym. Choć współczesne komputery oferują w pełni graficzny interfejs, wiersz poleceń wciąż często pozostaje **najefektywniejszym i najnaturalniejszym sposobem komunikacji z systemem** oraz uruchamiania programów.

Terminal


linuxbabe@bionic: ~
gnome-terminal(1) General Commands Manual gnome-terminal(1)
NAME
gnome-terminal – is a terminal emulation application.
SYNOPSIS
gnome-terminal [-e, --command=STRING] [-x, --execute] [--window-with-profile=PROFILENAME] [--tab-with-profile=PROFILENAME] [--window-with-profile-internal-id=PROFILEID] [--tab-with-profile-internal-id=PROFILEID] [--role=ROLE] [--show-menubar] [--hide-menubar] [--geometry=GEOMETRY] [--working-directory=DIRNAME] [-?, --help]
DESCRIPTION
GNOME Terminal is a terminal emulation application that you can use to perform the following actions:
Access a UNIX shell in the GNOME environment.
A shell is a program that interprets and executes the commands that you type at a command line prompt. When you start GNOME Terminal, the application starts the default shell that is specified in your system account. You can switch to a different shell at any time.
Manual page gnome-terminal(1) line 1 (press h for help or q to quit)

Katalog domowy

Każdy użytkownik w systemie Linux ma przypisany własny katalog domowy.

To miejsce przeznaczone na wszystkie nasze dane, pliki konfiguracyjne i ustawienia osobiste.

Ze względu na częste odwoływanie się do tego katalogu wprowadzono skrót ~, który zawsze oznacza katalog domowy bieżącego użytkownika.

Przykład użycia:

```
cd ~adrianna
```

Katalogi i pliki: podstawowe pojecia

Podstawowe polecenia do pracy z katalogami w Linuxie:

`pwd` – wyświetla aktualny katalog roboczy
`tree` – pokazuje strukturę drzewa katalogów
`cd` – zmienia katalog roboczy:
`cd ..` – wychodzi o poziom wyżej
`cd` (lub `cd ~`) – przechodzi do katalogu domowego
`ls` – wyświetla zawartość katalogu. Ważne opcje:
`ls -a` – pokazuje ukryte pliki (nazwy zaczynające się od kropki)
`ls -l` – wyświetla szczegółowe informacje o plikach
`ls -R` – listuje katalogi rekurencyjnie, czyli wraz z podkatalogami
`mkdir nazwa` – tworzenie katalogu o nazwie *nazwa*
`rmdir nazwa` – usuwa pusty katalog o nazwie *nazwa*

Polecenie `man`

Polecenie `man` – dokumentacja w Linuxie

man pozwala przeglądać dokumentację wszystkich programów w systemie.

Przykład:

```
man ls
```

Aby wyjść z man'a, naciśnij `q`.

Aby wyszukać tekst, użyj `/` i wpisz szukane słowo.

Kolejny wynik wyszukiwania znajduje się pod przyciskiem `n`.

Kopiowanie plikow i katalogów

Polecenie `cp` – kopiowanie plików i katalogów

`cp ścieżka1 ścieżka2` – kopiuje plik z ścieżka1 do ścieżka2.

Istotne opcje:

`-r` – kopiowanie rekurencyjne katalogu wraz z całą zawartością

Znak **&** i praca w “tle”

Uruchamianie programów w terminalu i odblokowanie terminala

- Jeśli program zostanie uruchomiony w terminalu bez znaku **&**, terminal może zostać „zablokowany” do czasu zakończenia programu.

- Aby odblokować terminal, można użyć następujących skrótów:
  1. **`Ctrl + C`** – zamyka aktualnie uruchomiony program
  2. **`Ctrl + Z`** – zawiesza program i przywraca kontrolę nad terminalem

Zawieszony program można kontynuować w tle za pomocą komendy: **`bg`**

(**`bg`** – *background*) – program działa dalej, a terminal pozostaje dostępny do wprowadzania kolejnych poleceń.

Polecenie `rm` i `mv` w Linuxie

`rm` – usuwanie plików i katalogów

`rm nazwa_pliku` – usuwa plik o podanej nazwie

Istotne opcje:

`-r` – rekurencyjnie (usuwa katalog wraz z całą zawartością)

`-f` – wymusza usunięcie pliku bez ostrzeżeń (UWAGA – nieodwracalne!)

`mv` – przenoszenie i zmiana nazwy plików

`mv ścieżka1 ścieżka2` – przenosi plik z położenia *ścieżka1* do *ścieżka2*

Może służyć do zmiany nazwy pliku w tym samym katalogu

Opcje:

`-f` – nadpisuje istniejący plik bez ostrzeżenia

Wzorce

Znaki specjalne w Linuxie (wildcards/globbing)

`*` – zastępuje dowolną liczbę dowolnych znaków

`?` – zastępuje dokładnie jeden dowolny znak

`[]` – określa zakres znaków, które mogą się pojawić

- `[abc]` – pasuje do a lub b lub c
- `[a-c]` – pasuje do liter od a do c
- `[0-9]` – pasuje do dowolnej cyfry
- `[!a-c]` – pasuje do dowolnego znaku poza a, b, c

`{}` – lista alternatyw oddzielonych przecinkami

`{numer1,numer2}` – pasuje do numer1 lub numer2

Wzorce

Przykłady użycia wildcardów w Linuxie

`*.txt` – wszystkie pliki kończące się na `.txt`

`file?.log` – pliki `file1.log`, `fileA.log`, ale nie `file12.log`

`[abc]*.sh` – pliki `.sh` zaczynające się od `a`, `b` lub `c`

`[0-9]*.csv` – pliki `.csv` zaczynające się od cyfry

`[!a-c]*.md` – pliki `.md`, które nie zaczynają się od `a`, `b` lub `c`

`{dane1,dane2}.txt` – pasuje do `dane1.txt` lub `dane2.txt`

Uprawnienia i użytkownicy

Podstawowa idea: W Linuxie każdy plik i katalog ma przypisane:

- **właściciela (user)**
- **grupę (group)**
- **zestaw uprawnień (permissions)**

Uprawnienia określają, kto i co może zrobić z danym plikiem.

Trzy typy uprawnień: Każdy plik ma trzy podstawowe prawa:

- **r (read)** – odczyt
- **w (write)** – zapis / modyfikacja
- **x (execute)** – wykonanie (np. uruchomienie programu)

Trzy poziomy dostępu: Uprawnienia są definiowane osobno dla:

- **u (user)** – właściciel pliku
- **g (group)** – grupa właściciela
- **o (others)** – pozostali użytkownicy

```
adrianna@MacBook-Pro pokaz% pwd
/Users/adrianna/pokaz
adrianna@MacBook-Pro pokaz% ls
silnia.py    witajcie     witka.py
adrianna@MacBook-Pro pokaz% ls -l
total 24
-rw-------@ 1 adrianna  staff  62  2 paź 20:32 silnia.py
-rwxr-xr-x@ 1 adrianna  staff  51  3 paź 09:11 witajcie
-rw-r--r--@ 1 adrianna  staff  60  2 paź 20:35 witka.py
adrianna@MacBook-Pro pokaz % ls -lh
total 24
-rw-------@ 1 adrianna  staff    62B  2 paź 20:32 silnia.py
-rwxr-xr-x@ 1 adrianna  staff    51B  3 paź 09:11 witajcie
-rw-r--r--@ 1 adrianna  staff    60B  2 paź 20:35 witka.py
adrianna@MacBook-Pro pokaz % ls -la
total 32
drwxr-xr-x   6 adrianna  staff   192  3 paź 09:16 .
drwxr-xr-x+ 56 adrianna  staff  1792  3 paź 09:09 ..
-rw-r--r--   1 adrianna  staff     4  3 paź 09:16 .ukryty_plik
-rw-------@  1 adrianna  staff    62  2 paź 20:32 silnia.py
-rwxr-xr-x@  1 adrianna  staff    51  3 paź 09:11 witajcie
-rw-r--r--@  1 adrianna  staff    60  2 paź 20:35 witka.py
adrianna@MacBook-Pro pokaz %
```

Uprawnienia i użytkownicy

`-rwxr-xr--`

Rozbicie:

`-` → typ pliku (zwykły plik)

`rwx` → właściciel: odczyt, zapis, wykonanie

`r-x` → grupa: odczyt, wykonanie

`r--` → inni: tylko odczyt

Najważniejsze polecenia:

`chmod` – zmiana uprawnień

`chown` – zmiana właściciela

`chgrp` – zmiana grupy

`ls -l` – podgląd uprawnień

Zapis numeryczny

Prawa dostępu do plików w Linuxie można zapisać za pomocą liczb całkowitych z zakresu 0–7. Każda liczba jest sumą wartości przypisanych poszczególnym uprawnieniom:

- `r` (read – odczyt) = 4
- `w` (write – zapis) = 2
- `x` (execute – wykonanie) = 1

Suma tych wartości daje jedną cyfrę:

- `7` = 4 + 2 + 1 → rwx
- `6` = 4 + 2 → rw-
- `5` = 4 + 1 → r-x
- `4` = 4 → r--
- `0` = brak uprawnień → ---

Polecenie chmod

`chmod` służy do zmiany praw dostępu do pliku lub katalogu.

Składnia: `chmod <uprawnienia> nazwa_pliku`

Tryb symboliczny

`chmod u+x,g+x,o+x plik.txt` -> Nadaje prawo wykonania (x) właścicielowi (u), grupie (g) i innym (o).

Tryb numeryczny (uprawnienia zapisuje się jako trzy cyfry (kolejno: właściciel, grupa, inni))

`chmod 744 nazwa_pliku`

Oznacza:

7 (rwx) – pełne prawa dla właściciela

4 (r--) – prawo odczytu dla grupy

4 (r--) – prawo odczytu dla pozostałych użytkowników

Wyświetlanie zawartości pliku tekstowego (.txt)

Do przeglądania plików tekstowych w Linuxie służą m.in. polecenia:

- `cat` – wyświetla całą zawartość pliku naraz
- `more` – wyświetla plik strona po stronie
- `less` – umożliwia wygodne przewijanie w przód i w tył

Przykłady użycia:

```
cat plik.txt
less plik.txt
more plik.txt
```

`less` jest najwygodniejsze przy dużych plikach, ponieważ pozwala swobodnie nawigować po treści.

Standardowe wejście i wyjście

Znaki wpisywane z klawiatury trafiają do programu przez tzw. standardowe wejście (standard input, stdin).

Odpowiedź programu pojawia się na standardowym wyjściu (standard output, stdout), które domyślnie jest wyświetlane na ekranie terminala.

Operatory < i | w Linuxie

< – przekierowuje standardowe wejście (stdin) z pliku.

Program zamiast danych z klawiatury otrzymuje zawartość wskazanego pliku.

```
program < plik.txt
```

| (pipe) – przekierowuje standardowe wyjście (stdout) jednego programu na standardowe wejście (stdin) drugiego programu.

```
polecenie1 | polecenie2
```

Operator | pozwala łączyć polecenia w tzw. potoki (pipelines), tworząc z nich jeden ciąg przetwarzania danych.

Operatory **>** i **>>** – przekierowanie wyjścia

Dane ze standardowego wyjścia (stdout) można zapisać do pliku przy użyciu operatorów **>** lub **>>**.

Różnica między nimi:

> – tworzy nowy plik i zapisuje do niego wynik działania programu.

Jeśli plik już istnieje, zostaje nadpisany.

>> – działa podobnie, ale jeśli plik istnieje, wynik zostaje dopisany na końcu pliku.

Przykład:

```
ls -l > plik.txt
```

➡ wynik polecenia `ls -l` zostanie zapisany do pliku `plik.txt`

(jeśli plik istniał wcześniej, jego zawartość zostanie zastąpiona).

Przykład użycia operatora | (potok)

Rozważmy polecenie:

```
ls -R | grep owca | less
```

Jak to działa?

`ls -R` – przeszukuje aktualny katalog rekurencyjnie i wypisuje wszystkie pliki oraz podkatalogi.

`grep owca` – filtruje otrzymane dane, wybierając tylko te linie, które zawierają słowo „owca”.

`less` – wyświetla przefiltrowany wynik strona po stronie, umożliwiając wygodne przeglądanie.

Idea potoku: Każde kolejne polecenie przetwarza wynik poprzedniego.

Polecenie `grep`

`grep` służy do wyszukiwania w pliku tekstowym wierszy zawierających podany wzorzec i wyświetlania ich na ekranie.

Przykład 1 – wyszukiwanie bez rozróżniania wielkości liter:

```
grep –i abc nazwa_pliku
```

Wyszukuje w pliku nazwa_pliku wszystkie linie zawierające ciąg znaków abc, niezależnie od tego, czy zapisany jest małymi czy wielkimi literami.

Przykład 2 – użycie wyrażeń regularnych:

```
grep 'A[lg]a' nazwa_pliku
```

Znajduje linie zawierające wyraz „Ala” lub „Aga”.

Wyrażenie `[lg]` oznacza, że w tym miejscu może wystąpić litera `l` lub `g`.

Archiwizacja plików `tar`

`tar` służy do łączenia wielu plików i katalogów w jeden zbiorczy plik (tzw. archiwum). Takie archiwum można następnie skompresować, np. programem `gzip`.

Podstawowa składnia:

```
tar -cvf nazwa_pliku.tar nazwa_katalogu
```

Znaczenie opcji:

`-c` – tworzy nowe archiwum (create)

`-v` – wyświetla przetwarzane pliki (verbose)

`-f` – określa nazwę pliku archiwum (file)

Powstaje plik `nazwa_pliku.tar` zawierający wskazany katalog wraz z całą zawartością.

Wypakowanie plików `tar`

`tar` pozwala również wypakować pliki z wcześniej utworzonego archiwum `.tar`.

Składnia:

```
tar -xvf nazwa_pliku.tar
```

Znaczenie opcji:

`-x` – wypakowuje archiwum (extract)

`-v` – wyświetla listę przetwarzanych plików (verbose)

`-f` – wskazuje nazwę pliku archiwum (file)

Po wykonaniu polecenia zawartość archiwum zostanie przywrócona w bieżącym katalogu.

Kompresja i dekompresja archiwów `gzip`

`gzip` służy do kompresji i dekompresji plików, np. archiwów `.tar`.

Kompresja:

```
gzip nazwa_pliku.tar
```

Tworzy plik `nazwa_pliku.tar.gz`

Oryginalny `plik .tar` zostaje usunięty po kompresji

Dekompresja:

```
gzip -d nazwa_pliku.tar.gz
```

Przywraca oryginalny `plik .tar`

Dekompresja i rozpakowywanie archiwów skompresowanych `tar`

Pliki typu `.tar.gz` można jednocześnie rozpakować i zdekompresować za pomocą polecenia `tar`.

Składnia:

```
tar -xvzf nazwa_pliku.tar.gz
```

Znaczenie opcji:

`-x` – wypakowuje archiwum (extract)

`-v` – wyświetla listę przetwarzanych plików (verbose)

`-z` – dekompresuje plik gzip (gzip)

`-f` – wskazuje nazwę pliku archiwum (file)

➡ Po wykonaniu polecenia zawartość archiwum zostaje przywrócona w bieżącym katalogu.

Nie tylko GNU/Linux

W systemie **MS Windows** konsolę można uruchomić za pomocą opcji „Uruchom”, wpisując polecenie **`cmd`** (skrót od command line). Polecenia dostępne w powłoce Windows różnią się od poleceń znanych z systemów uniksowych, ponieważ wywodzą się z systemu MS-DOS.

Z kolei **macOS** — począwszy od wersji X — jest w istocie pełnoprawnym systemem uniksowym. Aplikacja „Terminal” stanowi jego integralną część i umożliwia korzystanie z powłoki zgodnej z tradycją systemów Unix.

Bezpieczeństwo w Linuxie

1. Model wieloużytkownikowy

- Oddzielne konta użytkowników
- Administrator (root) ma pełne uprawnienia
- Ograniczenie dostępu minimalizuje ryzyko błędów

2. System uprawnień do plików

- r (odczyt), w (zapis), x (wykonanie)
- Podział na: właściciela, grupę, innych
- Dodatkowe mechanizmy: setuid, setgid, sticky bit

3. Izolacja procesów

- Każdy program działa jako osobny proces
- Ograniczony dostęp do zasobów systemu

4. Bezpieczne zarządzanie oprogramowaniem

- Instalacja z podpisanych repozytoriów
- Regularne aktualizacje bezpieczeństwa

5. Dodatkowe mechanizmy

- Zapora sieciowa (firewall)
- SELinux / AppArmor
- Szyfrowanie dysków

Wniosek:

Linux jest bezpieczny dzięki kontroli dostępu, izolacji użytkowników i centralnemu zarządzaniu systemem.

Logowanie na komputery wydziałowe

```
ssh -X login@tempac.okwf.fuw.edu.pl
```

do wpisania w terminalu komputera domowego (konieczność posiadania systemu z terminalem, ewentualnie wirtualnej maszyny lub programu **PuTTY klient SSH i terminal dla Windows**)

- **PuTTY** to popularny **klient terminalowy** dla systemu Windows, używany do łączenia się z serwerami Linux/Unix.
- Obsługuje protokół: **SSH** (*Secure Shell*).
- Pozwala m.in. na:
  - bezpieczne zdalne logowanie do serwera,
  - uruchamianie poleceń w trybie tekstowym.
- Interfejs PuTTY jest prosty: wpisujemy adres serwera, port, typ połączenia i logujemy się do systemu (instrukcja obsługi na stronie wydziału: https://www.fuw.edu.pl/siec-studencka.html)

Dziękuję za uwagę!

