

# POWTÓRZENIE WIADOMOŚCI Z SYSTEMÓW OPERACYJNYCH I SIECI KOMPUTEROWYCH - dział „PODSTAWOWE OPERACJE W SYSTEMIE LINUX”

Temat: **Przegląd poleceń powłoki systemu.**

## 1. Wybrane polecenia powłoki systemu:

### 1.1. Pomoc (help oraz man)

Aby poznać szczegóły dotyczące składni polecenia wystarczy wpisać w konsoli:

`nazwa_polecenia --help`

np. `locate --help`

Rozbudowany opis poleceń można znaleźć w podręczniku systemowym (manual) za pomocą polecenia:

`man nazwa_polecenia`

np. `man cd`

### 1.2. Pamięć poleceń – history

**history** – wyświetla pełną listę poleceń wydanych przez użytkownika

Argumentem polecenia może być liczba ostatnio wydanych poleceń, które pragniemy przejrzeć np.

`history 25` (wyświetlona zostanie lista 25 ostatnio wydanych poleceń)

1.3. **date** – wyświetla datę i czas

1.4. **cal** – wyświetla kalendarz

1.5. **df** – podaje ilość wolnej przestrzeni na dysku

1.6. **who** – wyświetla wykaz zalogowanych użytkowników systemu

1.7. **whoami** – sprawdzenie swojej nazwy użytkownika

1.8. **last** – pokazywanie ostatnio zalogowanych użytkowników

1.9. **du** – ustalenie ile miejsca zajmuje plik lub katalog

Aby zorientować się ile miejsca zajmuje dany plik czy katalog, możemy skorzystać z polecenia `du`.

np. `du katalog1`

Podawana w wyniku jego wykonania ilość miejsca jest prezentowana w jednostkach dyskowych. Wyświetlenie informacji w bardziej przystępny sposób odbywa się za pomocą polecenia `du` z parametrami.

np. `du -b kat1` – wyświetla liczbę bajtów, która dany element zajmuje na dysku.  
np. `du -s -b kat2` wyświetla całkowitą objętość danego elementu bez podawania zbędnych informacji.

**1.10. free** – wyświetla informacje o pamięci systemowej (ile RAM ma komputer)

**1.11. su** – przełączanie się na konto innego użytkownika

*Przykład:*

`su root` – przełączenie się na konto użytkownika root

Po poleceniu `su` wpisujemy login użytkownika, na które konto chcemy się przełączyć. Potem system pyta nas o hasło tego użytkownika.

### 1.12. Uzyskiwanie informacji o sprzęcie

Aby dowiedzieć się więcej o komputerze, na którym pracujemy możemy posłużyć się dwoma poleceniami: **`arch`** i **`uname`**. Oba służą do wypisywania informacji o typie sprzętu komputerowego.

- **`arch`** – wyświetla informację o architekturze komputera czyli rodzaju zastosowanego procesora
- **`uname`**

Wydanie polecenia `uname` bez parametru spowoduje wyświetlenie tylko informacji o używanym systemie operacyjnym.

`uname -a` – wypisanie wszystkich informacji o systemie

`uname -p` – wypisuje typ procesora

`uname -v` – wypisuje wersję systemu operacyjnego zainstalowanego na komputerze

**1.13. startx** – uruchamia powłokę graficzną



### Temat: Praca z plikami.

W Linuksie inaczej niż w systemie Windows nie jest wymagane stosowanie w nazwach plików specjalnych rozszerzeń.

Nazwy plików (do 255 znaków w nazwie)

Nazwy plików (i katalogów) nie powinny rozpoczynać się od znaku kropki. Znak kropki na początku nazwy pliku jest zarezerwowany dla ukrytych plików i katalogów .

**Kropka z przodu** – plik ukryty (*hide*) np. `.bash`

Nazwa nie może zawierać następujących znaków: `/, -, <, >, *, ?, [, ], &, ;`

Niezwykle istotne jest także wielkość stosowanych liter. Wielkie i małe litery są rozpoznawane jako osobne znaki.

## Znaki specjalne

? zastępuje jedną literę, znak

\* zastępuje dowolny ciąg znaków

np. ls c\* - odnajdź wszystkie pliki zaczynające się od litery c

ls ??? – wyświetl wszystkie pliki o trzyliterowej nazwie

### 1.1. Tworzenie plików

a) Zakładanie pliku z klawiatury – polecenie **cat**

Składania polecenia ma postać: **cat > nazwa zakładanego pliku**

gdzie > przekierowanie (przekierowanie na wyjście – ekran)

< przekierowanie na wejście (klawiatura)

>> (operator dopisywania) – wyjście diagnostyczne (ekran) dla błędów.

*Przykład:*

Założ w katalogu bieżącym plik o nazwie adresy.txt następującej treści

Kowalska Maria, ul. Kwiatowa 4 m. 1, 25-000 Kielce

Makarski Leon, ul. Królewny Snieżki 5 m. 23, 02-571 Warszawa

Etap pierwszy – utworzenie pliku o konkretnej nazwie

**cat > adresy.txt**

Etap drugi – wpisanie treści w pliku

Etap trzeci – domknięcie pliku

Wcisnąć kombinację klawiszy **Ctrl + D**

b) utworzenie pliku za pomocą edytora **vi**

Składania polecenia ma postać: **vi nazwa zakładanego pliku**



### 1.2. Wyświetlanie zawartości pliku

Aby wyświetlić zawartość pliku na ekranie używamy polecenia **cat**. (wyświetla ono cały plik). Polecenie to przyjmuje jako parametr plik lub listę plików oddzielonych znakami spacji i wyświetla je w takiej kolejności w jakiej zostały podane.

*Przykład:*

**cat plik1** – wyświetl zawartość pliku o nazwie plik1

Do wyświetlania plików ale **strona po stronie** służy polecenie **more**.

np. **more plik1**



### 1.3. Usuwanie plików

Jeżeli chcemy usunąć plik, powinniśmy użyć do tego celu polecenia **rm** wraz z nazwą pliku jako parametrem.

Np. `rm nowy_plik`



### 1.4. Kopiowanie plików i katalogów

W systemie Linuks kopiowanie plików i katalogów odbywa się za pomocą polecenia **cp** wraz z odpowiednimi parametrami. Ogólna składnia tego polecenia określa element, który zamierzasz skopiować oraz cel w którym ma się znaleźć kopiowany element:

Składania polecenia ma postać: **CP nazwa pliku źródłowego nazwa pliku docelowego**

*Przykład:*

`cp /home/filmy/shrek.avi /home/kopie` - skopiuj plik shrek.avi do katalogu kopie



### 1.5. Przenoszenie plików lub katalogów oraz zmiana ich nazwy

W systemie Linuks można przenosić pliki i katalogi oraz zmieniać ich nazwy za pomocą jednego polecenia – **mv**.

Aby zmienić nazwę pliku, używamy tego polecenia, podając za nim stara nazwę istniejącego już pliku i tę, na którą chcemy ją zamienić.

np. `mv plik1 plik2`

Elementy przenosimy niemal tak samo; jedyna różnica polega na tym, iż pierwszą nazwą jest element który chcemy przenieść, a drugą element docelowy. Ten drugi musi istnieć fizycznie na dysku – w przeciwnym razie polecenie to zmienia nazwę pierwszego.

*Przykład:*

`mv /home/operator/roboczy /home/kopia` - przenieś katalog roboczy do katalogu kopia



**Temat: Praca z katalogami.**

**Katalog główny** (ang. *root directory*) – katalog w systemie plików nadrzędny dla wszystkich innych katalogów (i również plików). W systemach uniksowych oznaczamy przez ukośnik (*/*), a w systemach dosowych (Windows) przez odwrotny ukośnik (*\*).

## Prawidłowa ścieżka dostępu do katalogu (pliku) w systemie Linux.

W większości systemów (Unix, Linux, Mac OS) katalog jest reprezentowany przez ukośnik (ang. slash "/"), pełna ścieżka do pliku twierdza.avi wygląda następująco:

/home/ciapek/filmy/twierdza.avi

### 1.1. Tworzenie katalogów

Katalogi tworzymy za pomocą polecenia **mkdir**. Jako parametr podajemy nazwę nowego katalogu.

np. *mkdir katalog1* – utwórz katalog o nazwie katalog1



### 1.2. Wyświetlanie zawartości katalogów

Wyświetlanie zawartości katalogu (zwane inaczej listowaniem) można wykonać za pomocą polecenia **ls**.

Aby uzyskać więcej informacji na temat zawartości katalogu stosujemy polecenie **ls** wraz z parametrami np.

- ls – l (wyświetlanie danych szczegółowych)
- ls – a (wyświetla wszystkie pliki łącznie z ukrytymi)
- ls – t (sortowanie plików według czasu modyfikacji)
- ls – u (sortowanie według czasu dostępu)
- ls –c (sortowanie plików według czasu ich utworzenia)



### 1.3. Przechodzenie pomiędzy katalogami – zmiana katalogu

Do poruszania się w strukturze katalogów stosujemy polecenie **cd**.

Składania polecenia ma postać: **CD nazwa katalogu, który ma stać się bieżącym**

*Przykłady:*

**cd ..** – przejście o jeden poziom wyżej

**cd roboczy** – przejście do katalogu ROBOCZY

**cd /roboczy/www** – przejście do katalogu WWW, który jest podkatalogiem katalogu ROBOCZY.

**cd** – podane bez parametrów powoduje przejście do katalogu domowego



### 1.4. Usuwanie katalogów

Katalogi w systemie Linuks usuwamy za pomocą polecenia **rmdir**.

Składania polecenia ma postać: **rmdir nazwa usuwanego katalogu**  
**Można jedynie usuwać pusty katalog !!!.**

Pamiętajmy także, że nie możemy znajdować się w katalogu, który mamy zamiar usunąć.

*Przykład:*

**rmdir kat1** – usunięcie katalogu kat1



### 1.5. Kopiowanie katalogów

Patrz temat: praca z plikami.

### 1.6. Przenoszenie katalogów oraz zmiana ich nazwy

Patrz temat: praca z plikami.

### 1.7. Wyświetlanie nazwy bieżącego katalogu

Do wyświetlania aktualnej ścieżki stosuje się polecenie **pwd**. Wyświetla ono całą ścieżkę, w której się obecnie znajdujemy, zaczynając od katalogu głównego na dysku.

Składnia: **pwd**



Temat: **Prawa dostępu w systemie Linux.**



Prawa dostępu określają zakres operacji, które można wykonać na pliku bądź katalogu. Służą także do ograniczenia dostępu do pliku bądź katalogu dla konkretnej grupy użytkowników.

Prawa dostępu do plików podawane są trójkami dla trzech różnych kategorii użytkowników. Pierwsza trójka dotyczy **uprawnień właściciela pliku**, druga dotyczy **uprawnień grupy**, a trzecia obejmuje wszystkich **pozostałych użytkowników systemu**.

Zapamiętaj kolejność : właściciel, grupa i cała reszta (inni).

*Przykład:*

- *rw- rw- r--*

### **Symbole oznaczające typy elementów:**

- (zwykły myślnik) - plik

**d** – katalog

**l** – dowiązanie

**b** – specjalny plik blokowy

**c** – specjalny plik znakowy

## Rodzaje praw dostępu w przypadku plików:

**r** – (read) uprawnienie do odczytu  
**w** (write) – uprawnienie do zapisu  
**x** – uprawnienie do uruchamiania

## Rodzaje praw dostępu w przypadku katalogów:

**r** – do przeszukania zawartości  
**w** – do zmiany zawartości  
**x** – do wejścia do katalogu

Do wyświetlenia listy praw dostępu do danego pliku lub katalogu służy polecenie **ls -l**.

## Interpretacja praw dostępu

- *rw- rw- r--*

właściciel i grupa, do której należy mogą odczytywać i zapisywać zawartość pliku. Wszyscy pozostali użytkownicy systemu mogą jedynie podglądać zawartość pliku

*drwx r-x r-x*

właściciel katalogu ma prawo do jego przeszukania,, zmiany jego zawartości i wejścia do katalogu. Grupa ma prawo do wejścia do katalogu i przeszukania go. Także wszyscy inni użytkownicy mają prawo do wejścia do katalogu i przeszukania go.

## Nadawanie praw dostępu do plików i katalogów

Do zmiany praw dostępu do pliku (katalogu) służy polecenie **chmod**.

**chmod** – zmienia zestaw praw dostępu do pliku (katalogu)

*Przykłady:*

*chmod +x pokaz\_konta* – nadanie dla każdej z grup uprawnienie do uruchamiania

*chmod u+x, g+x pokaz* - ustawienie prawa do wykonywania dla użytkownika i grupy (plik staje się plikiem wykonywalnym dla jego właściciela i grupy)

*chmod ugoa+ -=rwx <plik>*

## Oznaczenia:

**u** - właściciel pliku; **g** - grupa; **o** - pozostali; **a** - wszyscy; **-**(minus) - zabiera; **+** - daje; **=** - czyni podane prawa jedyne prawa dla pliku.

*chmod u+x, u+r, u+x, g+r, g+x, o+r pokaz\_konta*

można także zapisać:

*chmod u=rwx, g=rx, o=r pokaz\_konta*

Można także ograniczać zestaw uprawnień (poprzez stosowanie znaku minusa):

`chmod -x pokaz` (zablokowanie uruchamiania pliku)

Przykłady:

`chmod a-w jakiś_plik` - zabiera wszystkim możliwość edycji tego pliku

`chmod g-x jakiś_plik` - zabiera grupie możliwość wykonania pliku

`chmod o+w jakiś_plik` - nadaje pozostałym możliwość edycji pliku

`chmod g=r jakiś_plik` - ustala, że grupa może jedynie czytać plik

Określanie praw dostępu można także zapisać w formacie **numerycznym**. Polega ona sumowaniu liczb odpowiadających dostępowi do danego elementu.

- 4 – r
- 2 – w
- 1 – x

Przykład:

`chmod 777 plik`

7 – otrzymywane jest na wskutek sumowania składowych; oznacza rwxrwxrwx; wszystkim użytkownikom w systemie zezwolono na podejmowanie wszystkich możliwych akcji.

### Zadanie

Ustaw prawa dostępu do pliku tak aby właściciel pliku miał prawa do zapisu, odczytu oraz uruchomienia, a grupa do której został przypisany ten plik, tylko prawa do jego odczytu.

A zatem:

- Właściciel –  $4 + 2 + 1 = 7$
- Grupa –  $4 = 4$
- Inni –  $4 = 4$

`chmod 744 plik`



Temat: **Archiwizacja w Linux. Kompresja i dekompresja plików.**

**Archiwizacja** – wykonanie kopii danych i przechowywanie ich w bezpiecznym miejscu, głównie na nośnikach zewnętrznych.

Archiwizacja jest podstawowym działaniem zabezpieczającym przez utratą danych.



### Archiwa i formaty kompresji

Formaty kompresji:

- tar (archiwum nie skompresowane)
- tar.gz (kompresja archiwum programem GZIP)
- tar.bz2 (kompresja archiwum programem BZIP2)

Program tar służy do archiwizowania plików. Program ten służy do gromadzenia plików w jeden łatwy do przenoszenia pakiet (łączy grupę plików w jeden).

Pod Windows **twórzmy** paczki **RAR**, natomiast pod Linuxem **TAR**.

**Program TAR standardowo nie kompresuje archiwum**, a jedynie archiwizuje (do pliku) katalogi z zapisem układow podkatalogów i ich zawartością.

### Tworzenie archiwum tar (opcja c)

**Składnia polecenia tar:**

**tar opcje nazwa\_archiwum lista\_plikow\_do\_archiwizacji**

np.

tar cvf paczka.tar /etc (zarchiwizuj w pliku paczka.tar zawartość całego katalogu /etc)

**Opis wybranych opcji:**

- f zapisanie archiwum do pliku o określonej nazwie
- x rozpakowanie archiwum
- c utworzenie nowego archiwum
- z kompresja (dekompresja) archiwum programem GZIPp
- j kompresja/dekompresja archiwum programem BZIP2
- v nakazuje wyświetlenie listy wydobytych z archiwum plików
- C zapis plików z archiwum do podanego katalogu

Opis wszystkich opcji jest dostępny po wprowadzeniu polecenia **man tar**

Wykonywanie archiwizacji jest mocno związane z pojęciem **kompresji plików**.

**Kompresja danych** – spakowanie archiwizowanych danych dzięki czemu zajmują one mniej miejsca na nośniku. Działaniem przeciwnym do kompresji jest **dekompresja**. Proces odtworzenia oryginalnych danych na podstawie ich postaci skompresowanej.

Jeśli dodamy do polecenia opcję z  
tar cvfz paczka.tar /etc

to pliki będą dodatkowo kompresowane przez gzip.

### Rozpakowywanie (dekompresja) archiwum tar (opcja x)

**tar -xvf paczka.tar (rozpakowanie archiwum tar)**

**tar -xvzf plik.tar.gz** (rozpakowanie archiwum tar skompresowanego z użyciem programu gzip)

tar xzvf <nazwa\_pakietu>.tar.gz

tar xjvf <nazwa\_pakietu>.bz

tar xjvf <nazwa\_pakietu>.tar.bz2

**Rozpakowywanie archiwum tar do podanego katalogu (opcja C):**

**tar xfC paczka.tar /test2**

### Tworzenie archiwum w KDE i GNOME

Archiwum można także utworzyć w środowisku KDE i GNOME. Do tworzenia archiwum w środowisku graficznym służy program **Ark**.

**Ark** – jest to aplikacją, która pozwala tworzyć i otwierać skompresowane archiwa w szeregu formatach: tar, gzip, bzip, zip. Ark dla systemu ze środowiskiem KDE stanowi domyślne narzędzie do archiwizacji.

Tworzenie archiwum w Ark

1. Otwórz menu KDE/Użytki/Archiwizacja/Ark
2. Wybierz Plik/nowy
3. Na ekranie Utwórz nowe archiwum wpisz nazwę archiwum w polu lokalizacja
4. W polu Filtr wybierz typ archiwum (najlepsza jest opcja archiwum po tarowaniu i spakowaniu gzipem)
5. Kliknij Zapisz
6. Aby dodać dane do archiwum, wybierz Akcja/dodaj plik albo akcja Dodaj folder

Z kolei domyślnym narzędziem do archiwizacji w powłoce GNOME jest program **File Roller**.



Temat: **Instalowanie programów pod Linuxem**



Programy przewidziane do zainstalowania w systemie Linux są najczęściej rozpowszechniane w postaci plików: **\*.rpm**, **\*.tar.gz** lub **\*.tar** nazywanych też **paketami**.

#### **RPM (Red Hat Packet Management)**

Pliki o rozszerzeniach **.rpm** zawierają wersje instalacyjne aplikacji dla systemów linuksowych.

Rozszerzenie **RPM** jest skrótem od nazwy programu **Red Hat Packet Manager**. Instalowanie programów zapisanych w plikach **\*.rpm** w większości przypadków przebiega automatycznie. Większość dystrybucji wyposażona jest w zestaw narzędzi mających ułatwić procedurę instalacji (w Mandrake jest to RpmDrake albo polecenie powłoki **urpmi**).

#### **Instalowanie programów z kodu źródłowego (w powłoce)**

Archiwa kodu źródłowego są zazwyczaj rozprawdane w postaci tzw. paczek programu tar – archiwów wykonanych poleceniem tar. Dzieje się tak ponieważ licencja GNU GPL wymaga rozprawdania wraz z programami ich kodu źródłowego.

1. Rozpakuj archiwum poleceniem tar  
np. **tar -xvzf paczka.tar.gz**
2. Przejdź do katalogu kodu źródłowego (do katalogu w którym rozpakowane zostało archiwum)  
Np. **cd paczka**
3. Zaloguj się jako superużytkownik  
Np. **su**
4. Skompiluj program wpisując polecenia:
  - **./configure** (wywołuje skrypt tworzący plik sterujący kompilacją - Makefile)
  - **./install**ewentualnie  
ewentualnie

- **make install** (kompilacja programu)

### Gdzie szukać oprogramowania – linki

- <http://rpmfind.net/> (wyszukiwarka pakietów RPM)
- <http://www.tuxfinder.com/> (wyszukiwarka zarchiwizowanych i skompresowanych plików kodu źródłowego)



### Temat: Instalowanie systemu SUSE Linux 10.

Pierwszą czynnością jest sprawdzenie, czy komputer na którym będzie instalowany system SUSE Linux 10, spełnia minimalne wymagania sprzętowe systemu:

Komponent	Wymagania
Procesor	Pentium I –IV; AMD Duron, Athlon, Athlon XP, Intel Celeron 233 MHz
Pamięć operacyjna (RAM)	Przynajmniej 128 MB, zalecane 256 MB
Przestrzeń dyskowa	Przynajmniej 500 MB, zalecane 2,5 GB
Wyświetlanie	Karta graficzna i monitor w standardzie VGA obsługujące rozdzielczość 640x480
Dla instalacji z CD-ROM	Napęd CD-ROM
Peryferia	Klawiatura, mysz

### Zalecane rozmiary partycji

#### Typ instalacji

- Minimalna 500 MB
- Standardowa graficzna (jedno środowisko pulpitu, OpenOffice, przeglądarka WWW) 2 – 2,5 GB
- Przechowywanie multimediiów (muzyka wideo) 2 GB
- Nagrywanie CD 1 GB
- Wszystkie powyższe opcje 8 GB

Przed zainstalowaniem systemu warto sprawdzić czy posiadany sprzęt jest obsługiwane przez SUSE linux.

- (procesory, drukarki, płyty główne, karty grafiki) <http://cdb.suse.de>
- drukarki: [linuxprinting.org](http://linuxprinting.org)

### Dwa systemy

Ostatnim problemem który należy przemyśleć przed rozpoczęciem instalacji systemu SUSE Linux jest co zrobić z aktualnie zainstalowanym systemem operacyjnym. Jeśli ktoś chce zachować system Windows musi stworzyć dodatkową partycje (rozmiar powyżej)

Przy instalowaniu SUSE do uruchamiania systemu Linux z możliwością wyboru pomiędzy Linuxem i Windows, SUSE Linux instaluje na partycji rozszerzonej pozostawiając Windows na partycji podstawowej.



### Jak podzielić dyski na partycje

SUSE Linux do właściwego funkcjonowania wymaga dwóch partycji:

- partycja główna (root) oznaczona ukośnikiem (/)
- partycja wymiany (swap), której Linux używa jako pamięci podręcznej.

**Partycja wymiany (SWAP)** – systemowa partycja występująca w systemach typu Unix. Służy do tymczasowego przechowywania danych. Wielkość partycji SWAP jest równa (lub stanowi wielokrotność) wielkości pamięci RAM zainstalowanej w komputerze. Program konfiguracyjny (YaST) dobierze wielkość partycji wymiany automatycznie podczas instalacji.

Np. komputer ma 256 RAM partycja wymiany 256 MB (lub nawet 512MB)

## Instalacja SUSE Linux

Instalacja systemu SUSE Linux jest prostym procesem. Zawarta w nim aplikacja YaST prowadzi użytkownika przez cały proces wykrywa i konfiguruje sprzęt i ogólnie sprawuje kontrolę nad wszystkim.

### Etapy instalacji

1. Aby rozpocząć instalację uruchom komputer z płyty CD. By było to możliwe BIOS komputera musi obsługiwać funkcje rozruchu z napędu CD-ROM. Musisz dostać się do BIOS-u i uaktywnić przeszukiwanie napędu płyt. Aby dostać się do BIOS-u, bezpośrednio po włączeniu komputera naciśnij klawisz wskazany w komunikacie, który pojawi się przy stracie (w BIOS-ie firmy Award jest to klawisz *Delete*). Po wciśnięciu klawisza *Delete* widzimy program odpowiedzialny za konfigurację BIOS-u. Korzystając z klawiszy strzałek zaznacz pozycję o nazwie *Advanced BIOS Features*, a następnie naciśnij klawisz Enter. Widzimy okno konfiguracji zaawansowanych funkcji BIOS-u, a w nim kolejno: pierwszy, drugi i trzeci napęd, w którym komputer szuka systemu operacyjnego. Zaznacz opcje *First BOOT Device* i ustaw (za pomocą klawiszy *Page Up* lub *Page down*) *First Boot Device* – *CDROM*. Dzięki temu komputer uruchamia system operacyjny z płyty. Wróć do głównego okna BIOS-u (klawisz *Esc*). Za pomocą klawiszy strzałek zaznacz polecenie o nazwie *Save & Exit Setup* co oznacza Zapisz i opuść program konfiguracyjny. Gdy wciśniesz klawisz *Y* i *Enter* komputer uruchomi się od nowa. Zanim to zrobisz, włóż do napędu płyt krążek instalacyjny Windows.
2. Po uruchomieniu programu instalacyjnego z CD-ROM pojawi się menu z kilkoma opcjami startowymi. Wybierz język Instalacji (Polski) a następnie Instalacja. YaST zidentyfikuje i zainicjuje sprzęt niezbędny do rozpoczęcia instalacji, a następnie wyświetli umowę licencja. Kliknij Tak, zgadzam Se na warunki umowy licencyjnej aby przejść do pierwszego ekranu instalacji.
3. Podział dysku na partycje
4. Instalacja pakietów oprogramowania
5. Tworzenie użytkownika root
6. Konfiguracja sieci i połączeń internetowych
7. Tworzenie konta zwykłego użytkownika
8. Konfiguracja sprzętu dla X Window System (kartka grafiki, drukarka, karta dźwiękowa)
9. Logowanie i wyłączenie systemu po raz pierwszy



## Temat: **Uruchamianie systemu. Program rozruchowy.**

Start systemu Linux obejmuje cztery etapy:

1. BIOS komputera uruchamia sprzęt
2. Przekazanie przez BIOS kontroli nad komputerem do programu rozruchowego (LILO lub GRUB)
3. Ładowanie jądra
4. Logowanie



**Program rozruchowy** (ang. **bootloader**) - służy do załadowania systemu operacyjnego do pamięci operacyjnej. Pełni także funkcję menedżera uruchamiania (pozwala wybrać system do uruchomienia). Program rozruchowy znajduje się na dysku twardym (w sektorze MBR), na dyskietce lub płycie CD-ROM.

**LILO** (**L**inux **L**Oader) to program rozruchowy Linuksa.

Inny program rozruchowy pod Linux – **GRUB** (**G**Rand **U**nified **B**ootloader).

### **Wybór menedżera startowego GRUB czy LILO?**

W narzędziu YaST moduł *Boot Loader Configuration* jest dostępny na stronie *System*. Można tu wybrać LILO i inne opcje.



## Temat: **Tworzenie skryptów powłoki.**



**Skrypty powłoki** to pliki tekstowe, które zawierają ciągi poleceń dla powłoki systemowej. Skryty są bardzo pomocnym rozwiązaniem kiedy istnieje konieczność wykonywania złożonych poleceń i poleceń, które są powtarzane okresowo. Skrypty powłoki to nic innego jak polecenia, które zapisane są w jednym pliku. Każda nowa linia to nowe polecenie.

Skrypty powłoki muszą zostać poprzedzone w pierwszym wierszu odpowiednią instrukcją odwołującą się do interpretera powłoki, której używamy.

```
#!/bin/bash
```

Dodatkowo plik taki musi mieć prawa do wykonywania (prawo X), które należy podać mu poprzez polecenie **chmod**.

W celu uruchomienia skryptu należy go odpowiednio wywołać.

### **1. Sposoby uruchomienia:**

**bash skrypt** (użycie powłoki i przekazanie do niej skryptu w formie argumentu)

**./ skrypt** (użycie znaku specjalnego – kropki i ukośnika).



## 2. Komentarze

Komentarz pełni jedynie funkcję informacyjną. Jego treść jest ignorowana. Komentarz w skryptach poprzedzony jest znakiem #

## 3. Wypisywanie tekstu na ekranie

Aby wypisać tekst na ekranie użytkownika po poleceniu echo deklarujemy tekst, który się pokaże po wywołaniu skryptu.

**echo** "To jest tekst"

**Zadanie 1.** Napisz skrypt wyświetlający napis: "Witam".

**Przykład skryptu:**

```
#!/bin/bash
echo "Witam"
```



## 4. Zmienne

Zmienne to elementy, które mogą przechowywać wartości. Wartość do zmiennej najlepiej wpisywać w cudzysłowach.

*zmienna*="tekst"

Przy wyświetlaniu wartości zapisanej zmiennej należy użyć znaku dolara „\$” przed zmienną, aby wyświetlenie zadziało.

**echo** \$zmienna

**Zadanie 2.** Napisz skrypt, który przypisze słowo *Linux* do zmiennej *system* oraz, słowo *super* do zmiennej *opinia*.

**Przykład skryptu:**

```
#!/bin/bash
system="Linux"
opinia="super"
echo "$system jest $opinia ."
```



Zmienne są oczywiście po to, aby je zmieniać. Tak więc można na nich wykonywać operacje (np. matematyczne).

**Zadanie 3.** Napisz skrypt, który obliczy sumę dwóch liczb. Liczby mają następujące wartości:  $x=1$ ,  $y=2$ .

**Przykład skryptu:**

```
#!/bin/bash
x=1
y=2
suma=$((x+y))
echo "Suma wynosi $suma"
```



Do zmiennych można także przypisywać wartości podawane przez użytkownika z klawiatury. Czyli przy uruchomieniu skryptu, możemy zostać poproszeni o podanie jakiejś wartości. Do tego służy polecenie:

**read ZMIENNA**

Po wywołaniu takiego polecenia, powłoka będzie oczekiwała, aż ze standardowego wejścia (czyli z klawiatury) zostanie podana wartość. Następnie podana wartość będzie przypisana do zmiennej ZMIENNA.

**Zadanie 4.** Napisz skrypt, który wczyta dwie liczby i wypisze ich sumę.

**Przykład skryptu:**

```
#!/bin/bash
echo "Podaj pierwsza liczba"
read x
echo "Podaj druga liczba"
read y
suma=$((x+y))
echo "Suma wynosi $suma"
```



**Zadanie 5.** Napisz skrypt, który wczyta od użytkownika jego imię, a następnie wyświetli je na ekranie.

**Przykład skryptu:**

```
#!/bin/bash
echo "Jak masz na imię"
read imie
echo "Witaj $imie"
```



## 5. Parametry

Do skryptu można także przekazywać parametry. Używa się ich podobnie jak zmienne, z takim wyjątkiem że nie można zmienić ich wartości. Każdy parametr posiada swój numer, np. numer 0 to nazwa skryptu.

### Parametry przekazywane do skryptów.

- **\$#** - ilość parametrów.
- **\$\*** - wszystkie argumenty jako jeden tekst "\$1 \$2".
- **\$@** - wszystkie argumenty jako ciąg oddzielnych łańcuchów tekstowych "\$1" "\$2".
- **\$0** - nazwa pliku.
- **\$1-\$9** - poszczególne parametry (pierwszy, drugi, trzeci itd.).
- **\$?** - kod zakończenia operacji.
- **shift n** - komenda służąca do przesuwania parametrów (10=>9, 9=>8 .. 1=>bye). Parametr „n” określa przesunięcie.

Przykład :

```
zad zenek mirek

W skrypcie pod poszczególnymi zmiennymi będą
znajdowały się następujące wartości :

$# - 2
$* - "zenek mirek"
$@ - "zenek" "mirek"
$0 - „zad”
$1 - "zenek"
$2 - "mirek"
$3-$9- ""
```

To jak wartości poszczególnych zmiennych wykorzystamy zależy od nas.

W skryptach rozróżniamy zmienne typu numerycznego i zmienne typu tekstowego.

### Parametry przekazywane do skryptów:

**\$\*** - wszystkie argumenty jako jeden tekst

**\$0** - parametr 0 (nazwa skryptu)

**\$1** - parametr nr1

...

**\$9** - parametr nr 9

**\$#** - ilość parametrów

**\$\$** - numer procesu

### Przykład skryptu:

```
#!/bin/bash
echo Nazwa skryptu: $0
echo Ilosc parametrow: $#
echo Parametry: $*
echo Parametr1: $1
echo Parametr2: $2
echo Numer Procesu: $$
```

Aby przypisać parametry do skryptu, wystarczy wypisać ich wartości w odstępach po wywołaniu skryptu. Przykład wywołania skryptu z parametrami:  
`./nazwa_skryptu Zenek Mirek`

## 6. Operatory porównujące

### Instrukcje warunkowe.

- `if ... then ... fi`
- `if ... then ... else ... fi`

#### Operatory porównujące.

- `-eq` - (*equal to*) - równe z, dla stringów (=).
- `-ne` - (*not equal to*) - różne, dla stringów (!=).
- `-gt` - (*great that*) - większe niż,
- `-ge` - (*great-equal*) - większe-równe,
- `-lt` - (*less than*) - mniejsze niż,
- `-le` - (*less-equal*) - mniejsze-równe,

## 7. Instrukcja warunkowa 'if'

Instrukcja ta wybiera pomiędzy warunkami, w zależności od tego, czy są spełnione, czy nie wykonuje odpowiednie czynności.

```
if wartość  
then  
zrób coś  
fi
```

#### **Przykład skryptu:**

```
#!/bin/bash  
if [1 = 1]  
then  
echo "Wartości są równe"  
fi
```

#### **Rozbudowana wersja instrukcji if**

```
if wartość  
then  
zrób coś  
else  
zrób coś innego  
fi
```

**Przykład skryptu:**

```
#!/bin/bash
if [ 1 < 2 ]
then
echo "1 jest mniejsze od 2"
else
echo "1 jest wieksze od 2"
fi
```



**Zadanie 6.** Napisz skrypt, który wczyta od użytkownika dwie liczby, a następnie sprawdzi czy podane liczby są równe czy też różne od siebie.

**Przykład skryptu:**

```
#!/bin/bash
echo "Podaj pierwsza liczba"
read a
echo "Podaj druga liczba"
read b
echo "Podales: $a i $b"

if [ "$a" = "$b" ]
then
echo "Wartości są równe"
else
echo "Wartości są różne"
fi
```



**Zadanie 7.** Napisz skrypt, który wczyta od użytkownika dwie liczby i wypisze większą z nich.

**Przykład skryptu:**

```
#!/bin/bash
echo "Podaj pierwsza liczba"
read a
echo "Podaj druga liczba"
read b
echo "Podales: $a i $b"

if [ "$a" -gt "$b" ]
then
echo "$a jest wieksza od $b"
else
echo "$a jest mniejsze od $b"
fi
```

### UWAGA!

- W rozwiązaniu zastosowano operator porównujący. Zamiast znaku > użyto operator **-gt**.



**Zadanie 8.** Napisz skrypt, który wczyta od użytkownika dwie liczby, a następnie sprawdzi czy podane liczby są równe czy też różne od siebie (rozwiązanie za pomocą parametrów).

### Przykład skryptu:

```
#!/bin/bash
echo "Podales: $1 i $2"

if [ "$1" -eq "$2" ]
then
echo "Wartości są równe"
else
echo "Wartości są różne"
fi
```

### UWAGA!

- W rozwiązaniu zastosowano operator porównujący. Zamiast znaku = użyto operator **-eq**.
- Aby przypisać parametry do skryptu, wystarczy wypisać ich wartości w odstępach po wywołaniu skryptu. Przykład wywołania skryptu z parametrami:  
*./nazwa\_skryptu 20 45*

przy czym:

20 – parametr pierwszy (\$1)

45 – parametr drugi (\$2).



**Zadanie 9.** Rozbuduj skrypt nr 8, tak aby sprawdzał jeszcze dodatkowo, która z liczb jest większa.

### Przykład skryptu:

```
#!/bin/bash
echo "Podales: $1 i $2"

if [ "$1" -eq "$2" ]
then
echo "$1 jest rowne z $2"
else
echo "Liczby są różne"
  if [ "$1" -gt "$2" ]
  then
echo "$1 jest wieksze od $2"
  else
echo "$1 jest mniejsze od $2"
  fi
fi
```

## UWAGA!

- W rozwiązaniu zastosowano tzw. **zagnieżdżanie**, stąd wewnątrz instrukcji warunkowej if znalazła się kolejna instrukcja if (zagnieżdżanie – pot. "umieściłem if w if-ie").



**Zadanie 10.** Napisz skrypt, który wczyta od użytkownika jego wiek, a następnie obliczy i wypisze rok urodzenia tej osoby.

Przykład skryptu:

```
#!/bin/bash
echo "Ile masz lat"
read wiek
zgadnij=${2009-$wiek}
echo "Urodziłeś się w $zgadnij roku"
```



## 8. Instrukcja case

Instrukcja case sprawdza, czy warunek ma odpowiednią wartość, a następnie przechodzi do odpowiedniego fragmentu kodu w programie.

```
case warunek in
odpowiedzi1)
Zrób coś 1
;;
esac
```

**Przykład skryptu:**

```
#!/bin/bash
wartosc=1
case $wartosc in
1)
echo "Liczba ma wartość 1"
;;
2)
echo "Liczba ma wartość 2"
;;
esac
```

**Zadanie 11.** Napisz skrypt z użyciem instrukcji case, który zrealizuje poniższe zadania:

- wyświetli ścieżkę dostępu, w której znajduje się skrypt
- wyświetli wszystkie pliki w tym katalogu
- wyświetli listę procesów użytkownika.

Wprowadź identyfikatory dla zadań:

- s – wyświetl ścieżkę dostępu w której znajduje się skrypt,

w – wyświetl wszystkie pliki w tym katalogu,  
p – wyświetl listę procesów użytkownika.

**Przykład skryptu:**

```
#!/bin/bash
echo "Proste menu"
echo "s – wyświetl ścieżkę dostępu w której znajduje się skrypt"
echo "w – wyświetl wszystkie pliki w tym katalogu"
echo "p – wyświetl listę procesów użytkownika"
echo "Wybierz opcje: s,w,p"
read wybor

case $wybor in
s)
echo "Jestes w katalogu $(pwd)"
;;
w)
echo "Teraz wypisze wszystkie pliki w tym katalogu"
ls -a
;;
p)
echo "Teraz wypisze wszystkie Twoje procesy"
ps -a
;;
esac
```

