18 Programy narzędziowe w środowisku tekstowym systemu Linux

ZAGADNIENIA

- Jak posługiwać się edytorem tekstu vi?
- Jak tworzyć skrypty powłoki?
- Na czym polega kompresja i archiwizacja?
- W jaki sposób tworzy się archiwa w systemie Linux?
- W jaki sposób skompresować i zdekompresować pliki i katalogi?
- Gdzie znaleźć publikacje elektroniczne dotyczące systemu Linux?

18.1. Edytor tekstu vi

Edytor vi jest dostępny we wszystkich klonach systemu Unix / Linux, a w większości systemów jest edytorem domyślnym. Niektóre programy, np. cron, wykorzystują vi, jako swój bazowy edytor opcji. Sposób obsługi vi jest różny od sposobu obsługi edytorów pracujących w środowisku Windows. Każdy użytkownik Linuksa powinien więc poznać vi przynajmniej w takim stopniu, aby mógł wykorzystać go do edycji plików konfiguracyjnych. Istnieje w zasadzie tylko jedna metoda nauczenia się obsługi – trzeba ćwiczyć w praktyce. Na potrzeby początkujących użytkowników vi opracowano samouczek, który w ciągu kilkudziesięciu minut pozwoli na przećwiczenie podstawowych możliwości edytora. Interaktywny samouczek programu vi w wersji angielskiej jest dostępny między innymi na stronie `http://www.openvim.com/tutorial.html` lub w wersji polskiej na stronie `http://jaki-linux.org/aplikacje/uzywaj-vim-a-jak-profesjonalista-cz-1/`. Aby poznać najczęściej używane, należy uruchomić samouczek i wykonać wskazane ćwiczenia.

Istnieje kilka wersji edytora vi. Najpopularniejsza i najbardziej rozbudowana jest wersja VIM, która ma też najwięcej opcji i możliwości, lecz zasady pracy w obu programach są podobne. Edytor vi pracuje w środowisku tekstowym; może być uruchomiony poleceniem `vi`. Jeżeli trzeba utworzyć nowy lub otworzyć istniejący plik, można od razu podać jego nazwę, np. `vi plik.txt`.

Na ekranie można wydzielić dwa obszary robocze: obszar poleceń (ostatnia linia na dole ekranu) i obszar edycji tekstu (pozostałe linie). Edytor vi ma dwa tryby pracy: tryb poleceń do wydawania poleceń określających, co chce się zrobić, oraz tryb edycji, w którym edytuje się tekst. Po uruchomieniu vi domyślnie znajduje się w trybie poleceń. Aby przejść do trybu edycji, należy nacisnąć klawisz [INS] (insert); aby przejść do trybu poleceń, należy nacisnąć klawisz [Esc].

Aby zakończyć pracę z vi, należy nacisnąć: [Esc] `:q!` – bez zapisu tekstu do pliku, [Esc] `:wq` – z zapisem tekstu do pliku.

Przemieszczanie się po tekście odbywa się za pomocą klawiszy kursorów w obu trybach pracy, a dodatkowo w trybie poleceń można użyć przycisków [h] – lewo, [j] – dół, [k] – góra, [l] – prawo. Liczba opcji, jakimi dysponuje vi, jest bardzo duża. Aby poznać najczęściej używane, należy uruchomić samouczek i wykonać wskazane ćwiczenia.

18.2. Skrypty powłoki

Skrypty są zwykłymi plikami tekstowymi, w których są zapisane polecenia zrozumiałe dla powłoki. Zadaniem powłoki jest przetłumaczenie ich na polecenia systemu. Aby przygotować skrypt, należy:

- stworzyć plik, w którym trzeba umieścić kod, np.:

```
touch naszskrypt.bat
```

- za pomocą dowolnego edytora tekstu wpisać do pliku kolejno wykonywane polecenia;
- nadać uprawnienia do wykonywania pliku dla uprawnionych użytkowników;
- uruchomić skrypt:

```
./naszskrypt.bat
```

gdzie symbol **./** oznacza, że skrypt znajduje się w bieżącym katalogu.

Przykładowy, bardzo prosty skrypt może się składać z następujących poleceń:

```
#!/bin/bash
#Tu jest komentarz.
echo „Hello World"
```

Pierwsza linia skryptu zaczynająca się od znaków **#!** ma szczególne znaczenie – wskazuje na rodzaj powłoki, w której skrypt ma być wykonany. Tutaj skrypt zawsze będzie wykonywany przez interpreter poleceń **/bin/bash**, niezależnie od tego, jakiego rodzaju powłoki używamy w danej chwili.

Znak **#** (hasz) oznacza komentarz – wszystko, co znajduje się za nim w tej samej linii, jest pomijane przez interpreter.

Trzecia linia spowoduje wydrukowanie na standardowym wyjściu *(stdout)*, czyli na ekranie, napisu: „**Hello World**".

W języku powłoki występują pewne słowa zastrzeżone, np.:

!	done	esac	function	Select	while	time
case	elif	Fi	If	Then	{	[
Do	else	for	in	until	}	]

Najczęściej używane polecenia w skryptach:

- **echo** – służy do wydrukowania napisu na standardowym wyjściu. Można też przekierować strumień danych do pliku;
- **read** – czyta ze standardowego wejścia pojedynczy wiersz;
- zmienne programowe (*program variables*) – zmienne definiowane samodzielnie przez użytkownika, np. zmienna=„wartość" (nie może być spacji przed i po „="). Do zmiennej można odwołać się przez podanie jej nazwy poprzedzonej znakiem $, np. dla zmiennej x może to wyglądać następująco: **echo $x**;
- zmienne specjalne (*special variables, specialparameters*) – to najbardziej prywatne zmienne powłoki. Są one udostępniane użytkownikowi tylko do odczytu (istnieją jednak wyjątki). Kilka przykładów zmiennych specjalnych:
  - **$0** – nazwa bieżącego skryptu lub powłoki;
  - **$1..$9** – parametry przekazywane do skryptu (uwaga! tu wyjątek – użytkownik może modyfikować ten rodzaj zmiennych specjalnych);
  - **$?** – kod powrotu ostatnio wykonywanego polecenia;

  - **$$** – PID procesu bieżącej powłoki;
- zmienne środowiskowe (*environment variables*) – definiują środowisko użytkownika dostępne dla wszystkich procesów potomnych, np.:
  - **$HOME** – ścieżka do katalogu domowego użytkownika;
  - **$USER** – login użytkownika;
  - **$HOSTNAME** – nazwa hosta użytkownika;
  - **$OSTYPE** – rodzaj systemu operacyjnego.
- instrukcja warunkowa **if** – sprawdza, czy warunek jest prawdziwy; jeśli tak, to wykonane zostanie polecenie lub polecenia znajdujące się po słowie kluczowym **then**. Instrukcja kończy się słowem **fi**. Składnia polecenia jest następująca:

```
if warunek
then
poleceniel
else
polecenie2
fi
```

- **test** – służy do sprawdzania warunków. Składnia polecenia:

  test wyrażeniel operator wyrażenie2,

  może też być zapisane w nawiasach kwadratowych:

  [wyrażeniel operator wyrażenie2]

  Między nawiasami a treścią warunku muszą być spacje, tak jak powyżej.
  Polecenie **test** zwraca wartość **0** (*true*), jeśli warunek jest spełniony, i wartość **1** (*false*), jeśli warunek nie jest spełniony, np. **test –e plik**
  Wybrane przykłady operatorów polecenia **test**:

  -e plik istnieje;
  = sprawdza, czy wyrażenia są równe;
  != sprawdza, czy wyrażenia są różne;
  –d wyrażenie istnieje i jest katalogiem;
  –r można czytać plik;
  –w można zapisywać do pliku;
  –x można wykonać plik;
  –lt mniejsze niż;
  –gt większe niż;
  –ge większe lub równe;
  –le mniejsze lub równe.
- instrukcja **case** – pozwala na dokonanie wyboru spośród kilku wzorców.
  Sprawdzana jest wartość zmiennej po słowie kluczowym **case** i porównywana ze wszystkimi wariantami po kolei. Jeśli dopasowanie zakończy się sukcesem, wykonane zostanie polecenie lub polecenia przypisane do danego wzorca. W przeciwnym wypadku zostanie użyte polecenie domyślne oznaczone gwiazdką „*". Składnia polecenia:

```
case zmienna in
„wzorzec1") polecenie1 ;;
„wzorzec2") polecenie2 ;;
„wzorzec3") polecenie3 ;;
*) polecenie_domyślne
esac
```

PRZYKŁAD 18.1

```
#!/bin/bash
echo „Podaj cyfrę dnia tygodnia"
read d
case „$d" in
„1") echo „Poniedziałek"  ;;
„2") echo „Wtorek" ;;
„3") echo „Środa"  ;;
„4") echo „Czwartek"  ;;
„5") echo „Piątek"  ;;
„6") echo „Sobota"  ;;
„7") echo „Niedziela"  ;;
*) echo „Nic nie wybrałeś"
esac
```

- pętla **for** – wykonuje polecenia zawarte wewnątrz pętli, na każdym składniku listy. Składnia polecenia:

```
for zmienna in lista
do
polecenie
done
```

PRZYKŁAD 18.2

```
for x in jeden dwa trzy
do
echo „To jest $x"
done
```

Pętla **for** jest bardzo przydatna w sytuacjach, gdy użytkownik chce wykonać jakąś operację na wszystkich plikach w danym katalogu. Na przykład, gdy trzeba uzyskać listę wszystkich plików o danym rozszerzeniu znajdujących się w pewnym katalogu, należy wpisać:

```
#!/bin/bash
for x in *html
do
echo „To jest plik $x"
done
```

- pętla **select** – wygeneruje z listy słów po **in** proste ponumerowane menu, każdej pozycji odpowiada kolejna liczba od 1 wzwyż. Poniżej menu znajduje się znak zachęty, gdzie wpisuje się cyfrę odpowiadającą wybranej przez użytkownika pozycji w menu. Jeśli nic się nie wpisze i zostanie wciśnięty [Enter], menu będzie wyświetlone ponownie. To, co zostało wpisane, jest zachowywane w zmiennej REPLY. Gdy odczytane zostaje EOF (*End Of File*), czyli znak końca pliku [Ctrl] + [D], to **select** kończy pracę. Pętla działa do momentu, gdy nie zostanie wykonane polecenie **break** lub **return**. Składnia polecenia:

```
select zmienna in lista
do
polecenie
done
```

Od razu nasuwa się możliwość zastosowania wewnątrz niej instrukcji **case**:

```
#!/bin/bash
echo „Co wybierasz?"
select y in X Y Z Quit
do
case $y in
„X") echo „Wybrałeś X"  ;;
„Y") echo „Wybrałeś Y"  ;;
„Z") echo „Wybrałeś Z" ;;
„Quit") exit ;;
*) echo „Nic nie wybrałeś"
esac
break
done
```

- pętla **while** – najpierw sprawdza warunek, czy jest prawdziwy; jeśli tak, to wykonane zostanie polecenie lub lista poleceń zawartych wewnątrz pętli. Jeżeli warunek jest fałszywy, pętla zostanie zakończona. Składnia polecenia:

```
while warunek
do
polecenie
done
```

PRZYKŁAD 18.3

```
#!/bin/bash
x=1;
while [ $x -le 10 ] do
echo „Napis pojawił się po raz: $x"
x=$[x + 1]
done
```

- pętla **until** – sprawdza, czy warunek jest prawdziwy; jeśli jest fałszywy, to wykonywane jest polecenie lub lista poleceń zawartych wewnątrz pętli między słowami kluczowymi **do** a **done**. Pętla **until** kończy swoje działanie w momencie, gdy warunek stanie się prawdziwy. Składnia polecenia:

```
until warunek
do
polecenie
done
```

PRZYKŁAD 18.4

```
#!/bin/bash
x=1;
until [ $x -ge 10 ] do
echo „Napis pojawił się po raz: $x"
x=$[x + 1]
done
```

18.3. Archiwizacja zbiorów systemu Linux

Podstawowym narzędziem do obsługi archiwów w Linuksie jest program tar. Normalnie **tar** tworzy nieskompresowane archiwum. W archiwum może znajdować się wiele plików i katalogów. Program **tar** domyślnie tworzy archiwum rekurencyjnie (z podkatalogami), umieszczając w nim wszystko, co znajdzie we wskazanym katalogu (w tym pliki i katalogi ukryte).

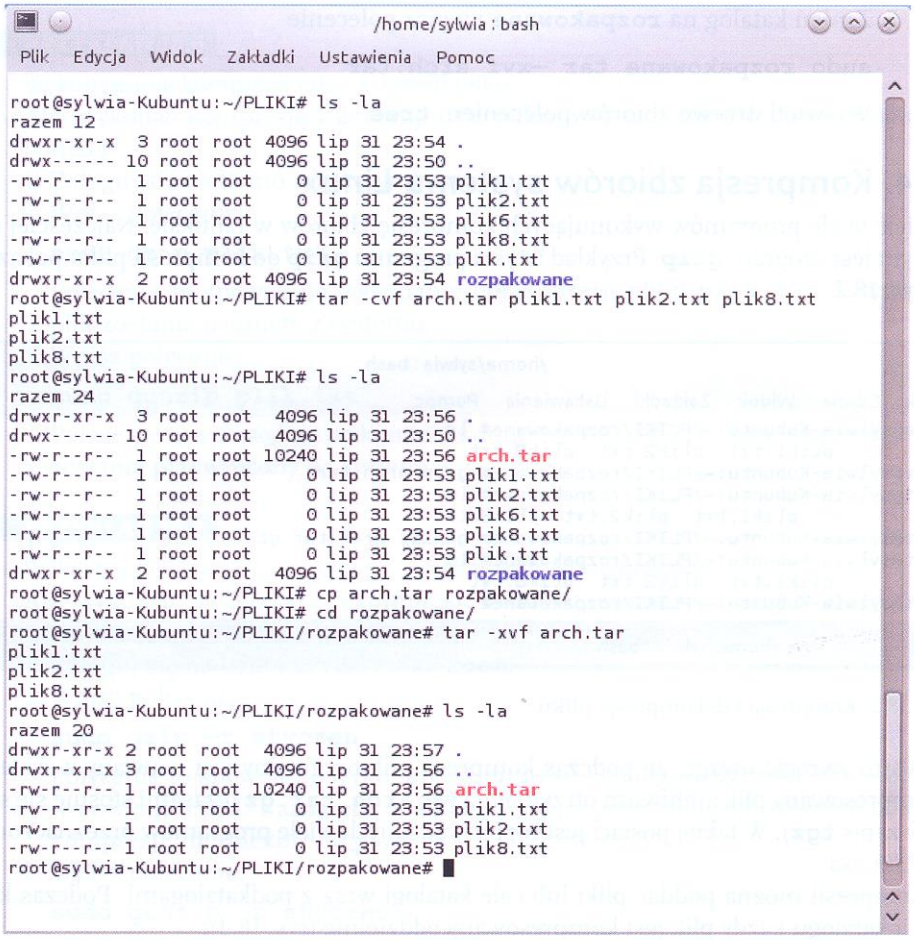
Składnia polecenia **tar** jest następująca:

```
tar opcje nazwa_archiwum plik
```

Najczęściej używane opcje polecenia **tar** to:

- **c** – tworzenie archiwum;
- **v** – podczas przetwarzania archiwum wyświetlane będą nazwy zbiorów;
- **f** – użycie wskazanego pliku jako archiwum;
- **x** – wyodrębnienie zbiorów z archiwum.

Przykłady obsługi archiwum pokazano na rys. 18.1.



Rys. 18.1. Polecenia obsługujące archiwizacje i wyodrębnianie zbiorów

PRZYKŁAD 18.5

Tworzenie archiwów zawierających zbiory

Aby utworzyć archiwum zawierające strukturę zbiorów utworzonych w przykładzie 18.1, wykonaj czynności przedstawione poniżej.

1. Zaloguj się na konto użytkownika **root**.
2. W katalogu domowym wpisz polecenie

   ```
   sudo tar -cvf arch.tar styczen/
   ```

   Polecenie to utworzy archiwum zawierające wszystkie zbiory z katalogu **styczen**. Podczas przetwarzania katalogu będą wyświetlane nazwy aktualnie przetwarzanego zbioru.
3. Utwórz katalog, w którym archiwum zostanie rozpakowane poleceniem

   ```
   sudo mkdir rozpakowane
   ```

4. Skopiuj plik **arch.tar** do katalogu **rozpakowane** poleceniem

   ```
   sudo cp arch.tar rozpakowane/arch.tar
   ```

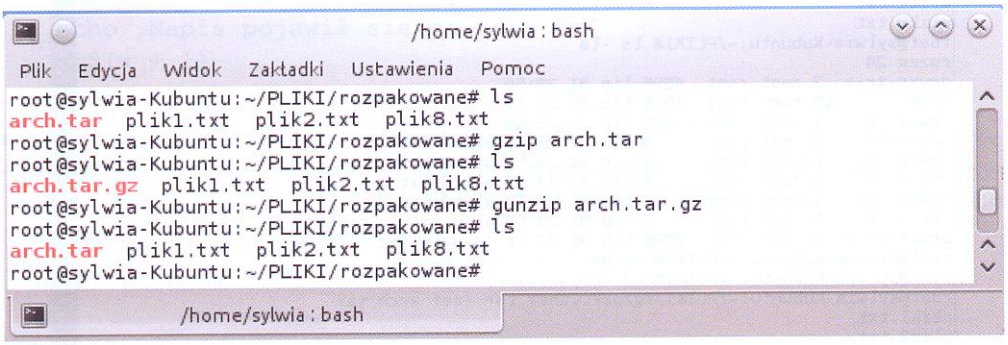
5. Zmień katalog na **rozpakowane** i wpisz polecenie

   ```
   sudo rozpakowane tar -xvf arch.tar
   ```

6. Wyświetl drzewo zbiorów poleceniem **tree**.

18.4. Kompresja zbiorów systemu Linux

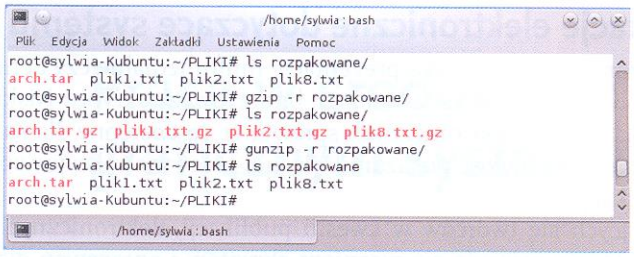
Istnieje wiele programów wykonujących kompresję zbiorów w Linuksie. Najczęściej używanym jest program **gzip**. Przykład użycia programu **gzip** do kompresji pliku pokazano na rys. 18.2.



Rys. 18.2. Kompresja i dekompresja pliku

Warto zwrócić uwagę, że podczas kompresji plik oryginalny jest usuwany z systemu. Skompresowany plik archiwum otrzymał nazwę **arch.tar.gz** (czasami stosuje się skrócony zapis **tgz**). W takiej postaci jest rozprowadzanych wiele programów przeznaczonych dla Linuksa.

Kompresji można poddać pliki lub całe katalogi wraz z podkatalogami. Podczas kompresji katalogu każdy plik jest kompresowany oddzielnie (rys. 18.3).



Rys. 18.3. Kompresja i dekompresja katalogu

W Linuksie **tar** może dokonać dodatkowej kompresji utworzonego archiwum, korzystając z programu **gzip**, w takiej sytuacji należy użyć dodatkowej opcji **-z**. Przykład użycia polecenia tworzącego skompresowane archiwum za pomocą programu **tar**:

```
sudo tar -czvf arch.tgz katalog
```

Do dekompresji użyjemy polecenia:

```
sudo tar -xzvf arch.tgz rozpakowane
```

PRZYKŁAD 18.6

Wykonywanie kompresji i dekompresji pliku

Aby wykonać kompresję i dekompresję pliku, wykonaj czynności przedstawione poniżej.

1. Zaloguj się na konto użytkownika **root**.
2. Wpisz polecenie:

   ```
   sudo gzip plik.txt
   ```

   Polecenie to spowoduje wykonanie dekompresji pliku **plik.txt**. Oryginalny plik zostanie usunięty z systemu.
3. Wpisz polecenie:

   ```
   sudo gunzip plik.txt
   ```

   Polecenie to spowoduje wykonanie kompresji pliku **plik.txt**. Oryginalny plik zostanie przywrócony do systemu.

PRZYKŁAD 18.7

Wykonywanie kompresji i dekompresji katalogu zawierającego pliki

Aby wykonać kompresję i dekompresję katalogu, wykonaj czynności przedstawione poniżej.

1. Zaloguj się na konto użytkownika **root**.
2. Wpisz polecenie:

   ```
   sudo gzip -r styczen
   ```

   Polecenie to spowoduje wykonanie kompresji katalogu **styczen**, utworzonego w ćwiczeniu 18.1. Każdy plik w katalogu i jego podkatalogach zostanie poddany kompresji, niezależnie od pozostałych zbiorów.
3. Wpisz polecenie:

   ```
   sudo gunzip -r styczen
   ```

   Polecenie to spowoduje wykonanie dekompresji plików w katalogu **styczeń** i jego podkatalogach.

18.5. Publikacje elektroniczne dotyczące systemu Linux

Publikacje elektroniczne to nie tylko prezentacje multimedialne czy demonstracje. Coraz więcej firm i osób publikuje własne katalogi, biuletyny czy skrypty. Niski koszt produkcji i dystrybucji wydawnictw elektronicznych sprawił, że są dostępne dla każdego. Dla wielu twórców pojawiła się możliwość pokazania własnej pracy w atrakcyjny sposób i to od razu na bardzo dużą skalę.

Do wyróżniających się twórców w kwestii publikacji elektronicznych należy zaliczyć wszystkich tych, którzy zajmują się pisaniem skryptów i opracowań dla systemu Linux. Ogromną bazę wiedzy zamieszczono na stronie **http://linux.skryptoteka.pl/**, gdzie znajdziemy obszerne publikacje oraz wersje elektroniczne pozycji książkowych, obszerną dokumentację Linuksa z podziałem na kategorie i podręczniki najpopularniejszych dystrybucji.

Również sam system wyposażony jest w obszerną pomoc systemową udostępnianą w postaci podręcznika.

Podobnie jak dla systemu Windows, na stronie **Dobre programy** możemy wyszukać ciekawe demonstracje dotyczące konfiguracji i pracy w systemie Linux.

Liczne informacje uzyskamy również na stronie **http://www.ubuntu-pomoc.org/**. Tam też znajdziemy wiele porad praktycznych i sprawdzonych rozwiązań w pracy z systemem.