PODSTAWY SYSTEMÓW OPERACYJNYCH
laboratorium nr 7

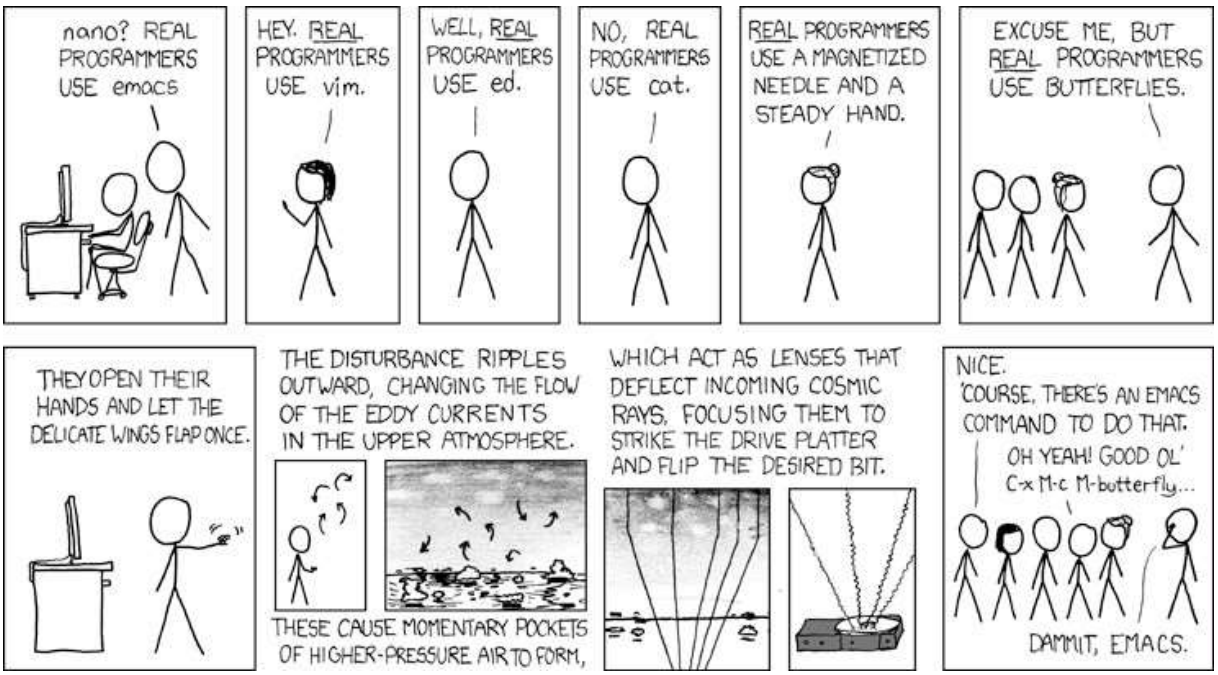
Wprowadzenie do skryptów Bash

PRACA ZALICZENIOWA NA PODSTAWY SYSTEMÓW OPERACYJNYCH

Celem pracy jest zawarcie wszystkich niezbędnych informacji potrzebnych do rozpoczęcia pracy ze skryptowym językiem programowania Bash dostępnym w środowisku Linux. Dokument oparty jest na
książce „Podstawy open source – system SUSE Linux cz. II" Novell Inc. 2008 poleconej przez wykładowcę, wzbogacony o aktualne źródła wiedzy dostępne w Internecie oraz własne doświadczenie. Cała ta wiedza została zorganizowana w przejrzysty sposób, a przykłady programów przetestowane i zaadaptowane, tak aby była to przystępna pomoc naukowa dla studentów.

Marcin Kulczak, Informatyka Stosowana WTiE UTP Bydgoszcz

Edytory tekstu



1. **VIM – Vi IMproved**

Najpopularniejszy edytor tekstu, jego uboższa wersja vi jest preinstallowana we właściwie każdej dystrybucji.

Ta aplikacja jest ulepszoną wersją edytora VI, który jest używany w systemie operacyjnym opartych na jądrze Linux jako standardowy program. Edytor VIM oferuje zaawansowaną funkcjonalność, zwiększoną moc i wiele innych opcji. Aplikacja została opracowana z uwzględnieniem wszystkich potrzeb programistów.

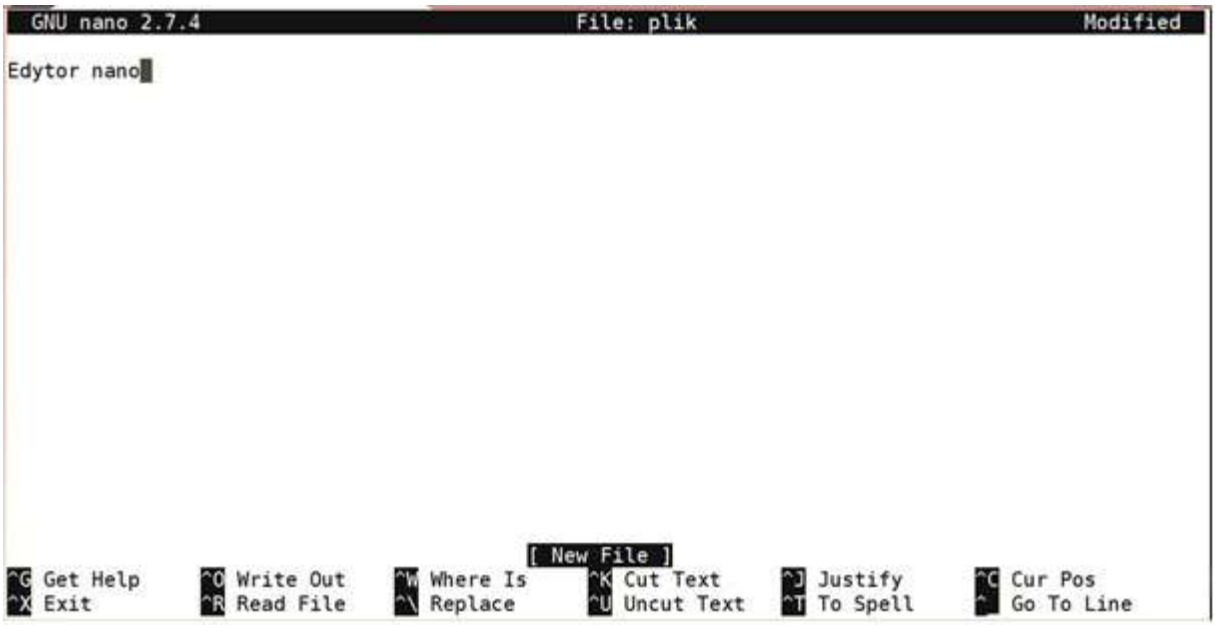
Ma ogromną liczbę ustawień, więc wśród użytkowników Linuksa jest często nazywany "edytorem dla programistów". Charakterystyczne cechy VIMa:

- zapewniony jest system znakowania
- istnieje możliwość rozszerzenia zakładki
- jest ekran sesji
- możesz podzielić ekran
- wprowadzane są różne symbole złożone

Instalacja:

```
apt-get update apt-
get install vim
```

2. Nano



Program Nano jest jednym z najpopularniejszych edytorów tekstu zaprojektowanych specjalnie dla platform UNIX. pierwsza wersja programu została opracowana w 2000 roku. Posiada ogromną liczbę dodatkowych funkcji, dzięki którym programiści uważają go za bardzo zaawansowany edytor kodu źródłowego i tekstu. Uwaga: Nano jest wyświetlane tylko w interfejsie wiersza poleceń.

Z powodu intuicyjnej obsługi, zaleca się nowym użytkownikom dystrybucji Linux na korzystanie właśnie z tego edytora.

Jeśli nano nie jest zainstalowane (a powinno być w standardzie), możemy go uzyskać poprzez komendę apt:

```
apt-get update apt-
get install nano
```

3. Emacs

Najbardziej rozbudowany program z całego zestawienia. Instalacja wraz z wszystkimi zależnościami zajmie około 200MB na dysku. Emacs charakteryzuje integracja z kompilatorami, systemami kontroli wersji oraz programem powłoki systemowej make. Dodatkowo daje możliwość podświetlania i automatycznego formatowania kodu źródłowego. Ponadto zapewnia możliwość pracy z interfejsem debuggera poprzez zainstalowanie ekskluzywnego rozszerzenia.

Edytor możemy zainstalować komendami:

```
apt-get update apt-
get install emacs
```

Utworzenie skryptu

Po zalogowaniu do terminala przejdźmy do naszego katalogu domowego.

```
cd ~
```

Utwórzmy plik skrypt.sh w edytorze nano. W tym celu:

```
nano skrypt.sh
```

Każdy skrypt musi zaczynać się od tzw. shebanga. Tę deklarację umieszczamy w pierwszej linii:

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

W uproszczeniu oznacza to, że skrypt ma być wykonany w powłoce bash.
Nasz pierwszy skrypt będzie wypisywał napis „Działa”. Po umieszczeniu shebanga, piszemy:

```
#!/bin/bash
echo "Działa"
```

Po wpisaniu wychodzimy z edytora naciskając CTRL+X.
Nano zapyta nas czy chcemy zapisać zmiany. Naciskamy y (lub t w przypadku spolszczonej wersji systemu).

Ostatnim krokiem przed uruchomieniem skryptu jest nadanie uprawnień wykonywania do pliku:
`chmod u+x skrypt.sh`

Wszystko gotowe. Wywołujemy nasze dzieło komendą:
`./skrypt.sh`

W terminalu powinien pojawić napis „Działa”.
Analogicznym sposobem należy tworzyć przyszłe skrypty.

Poznanie składni na klasycznych przykładach

1. Użycie *zmiennych*

```
#!/bin/bash

to_zmienna="Jan "
lokalizacja=$(pwd)
echo "Hello $to_zmienna!"
echo “Jestes w $lokalizacja”
```

Wynik:

```
Hello Jan !
“Jestes w /home/teleinspiro/bashowanie”
```

Wytłumaczenie:

Sposób deklarowania funkcji widoczny w pierwszej linii, nazwa zmiennej nie może zawierać białych znaków => spacji oraz nie może zaczynać się od cyfry. Aby użyć zmiennej, poprzedzamy ją znakiem $.

Aby podstawić do zmiennej wynik polecenia, komendę musimy ująć w nawiasach (linia nr 2). Dla powyższego przykładu obecnym katalogiem terminala był:
/home/teleinspiro/bashowanie, stąd taki wynik komendy pwd.

W dalszych przykładach tek sekcji pomijany jest shebang!

2. Cytowanie

```
NAME="Jan"
echo "Hej $NAME"  #=> Hej Jan
echo 'Hej $NAME'  #=> Hej $NAME
```

Wynik:

```
Hej Jan
Hej $NAME
```

Wytłumaczenie:

Dopisanie apostrofu do $NAME (co normalnie odniosłoby się do zmiennej) powoduje dosłowne wypisanie wyrażenia.

3. Funkcje

```
get_name() {
echo "Jan"
}

echo "Nazywasz się $(get_name)"
```

Wynik:

```
Nazywasz się Jan
```

Wytłumaczenie:

Na początku zadeklarowaliśmy funkcję o nazwie get_name, która ma na celu wypisanie łańcucha „Jan”. Aby z niej skorzystać, musimy napisać nazwę funkcji w nawiasach poprzedzając to wszystko znakiem dolara.

4. Brace expansion

```
echo {A,B}.js
echo {A,B}
echo {A,B}.js
echo {1..5}
```

Wynik:

```
A.js B.js
```

```
A B
```

```
A.js B.js
```

```
1 2 3 4 5
```

Wytłumaczenie:

Brace expansion pozwala na szybkie generowanie podobnych stringów, jak w przykładzie. Mechanizm ten może być osadzony => echo a{d,c,b}e da nam wynik:

ade ace abe

5. Instrukcje warunkowe

```
read string
if [ -z "$string" ]; then
echo "String jest pusty"
elif [ -n "$string" ]; then
echo "String nie jest pusty"
fi
```

Wynik:

Jeżeli po wykonaniu skryptu wpiszemy cokolwiek:

```
String nie jest pusty
```

Jeżeli po wykonaniu skryptu naciśniemy enter:

```
String jest pusty
```

Wytłumaczenie:

Pierwsza linia wczytuje input użytkownik do zmiennej string.
Następnie wykonuje się instrukcja warunkowa, gdzie w teście logicznym operatory:
-z : wyrażenie ma zerową długość
-n: wyrażenie ma długość większą od 0