Skrypty

Skrypty są odpowiednikami plików wsadowych systemu DOS. Skrypty są potężnym narzędziem programistycznym (rozbudowana składania, instrukcje pętli, warunku i wyboru, zmienne, stałe i wiele wiele innych) Skrypty czyli programy powłoki są prostymi plikami tekstowymi zawierającymi niektóre polecenia Linuksa i powłoki bash. Po uruchomieniu programu powłoki polecenia te są wykonywane jedno po drugim. Program powłoki może mieć przekazane parametry, tak jak w wypadku jakiegokolwiek normalnego polecenia. Parametry te mogą być oceniane wewnątrz programu. Ponieważ proste, sekwencyjne wykonywanie kilku poleceń nie zostawia wiele miejsca dla złożonych zadań, bash wspiera programowanie powłoki poleceniami do konstruowania pętli i rozgałęzień. Dzięki temu mamy do dyspozycji prawdziwy język programowania, do którego nie potrzebujemy wiedzy na temat kompilatora lub języka C (przypuszczalnie porównanie tego języka z C jest nadużyciem: programy C są wykonywane znacznie szybciej, obsługują więcej typów zmiennych, oferują więcej funkcji specjalnych itd. Niemniej jednak możliwości bash, są więcej niż wystarczające, aby poradzić sobie z zaskakująco dużą liczbą problemów). Typowym zastosowaniem programów powłoki jest automatyzacja najczęściej wykorzystywanych sekwencji poleceń do instalacji programów, administracji systemem, tworzenia kopii zapasowych, konfiguracji i wykonywania pojedynczych programów itd. Czyli podsumowując skrypt to niekompilowany tekstowy plik wykonywalny, zawierający jakieś polecenia systemowe oraz polecenia sterujące jego wykonaniem (instrukcje, pętle itp.). Wykonywany jest tylko i wyłącznie przez interpreter (tutaj /bin/bash), który tłumaczy polecenia zawarte w skrypcie na język zrozumiały dla procesora.

Pisanie skryptów

Aby napisać skrypt należy najpierw stworzyć jakiś plik w którym umieścimy kod. Do tworzenia pliku można użyć polecenia:
touch naszskrypt
Następnie za pomocą dowolnego edytora ASCII (mcedit, vi, itp.) należy wpisać do niego następującą zawartość:

```
#!/bin/bash
#Tu jest komentarz.
echo "Pierwszy program"
```

Znak # (hasz) oznacza komentarz, wszystko co znajduje się za nim w tej samej linii, jest pomijane przez interpreter. Pierwsza linia skryptu zaczynająca się od znaków: **#!** ma szczególne znaczenie - wskazuje na rodzaj shella w jakim skrypt ma być wykonany, tutaj skrypt zawsze będzie wykonywany przez interpreter poleceń **/bin/bash**, niezależnie od tego jakiego rodzaju powłoki w danej chwili używamy.

```
echo "Pierwszy program"
```

Wydrukuje na standardowym wyjściu (stdout) czyli na ekranie napis: Pierwszy program.

Aby móc uruchomić skrypt należy mu jeszcze nadać atrybut wykonywalności za pomocą polecenia:

```
chmod +x naszskrypt
```

Jeśli katalog bieżący w którym znajduje się skrypt nie jest dopisany do zmiennej PATH, to nasz skrypt możemy ururchomić w ten sposób:

```
./naszskrypt
```

Przykłady skryptów

Oto przykład bardzo prostego skryptu. Służy on do uruchamiania kilku poleceń.

```
#!/bin/bash
echo "Witam. Twoj login to $USER"
echo "Lista plików w bieżącym katalogu, $PWD"
ls # wypisz listę plików
```

Zauważ, jak wygląda komentarz w czwartej linii. W skrypcie dla bash-a wszystko stojące za znakiem # jest przez shell ignorowane. **$USER** oraz **$PWD** to zmienne. Są to standardowe zmienne zdefiniowane przez bash-a, więc nie muszą być definiowane oddzielnie w skrypcie. Podczas wykonywania skryptu nazwy zmiennych poprzedzone znakiem dolara są zastępowane przez ich aktualną zawartość. Nazywane jest to rozwijaniem zmiennych. Aby móc uruchomić skrypt należy mu nadać atrybut wykonywalności za pomocą polecenia:

```
chmod +x mójskrypt
```

Jeśli katalog bieżący w którym znajduje się skrypt nie jest dopisany do zmiennej PATH, to nasz skrypt uruchamiamy:

```
./mójskrypt
```

W tym przykładzie menu zrobione na instrukcji case wyświetla trzy opcje plus wyjście. Oparte jest na pętli until i funkcji.

```
#!/bin/bash
KONIEC="x";
function wybor
{
case $KONIEC in
s) echo "Jesteś w katalogu : $(pwd)" ;;
p) echo "teraz wypisze wszystkie twoje procesy";
ps -e ;;
w) echo "Oto lista plików w tym katalogu";
```

```
ls ;;
esac;
}

function menu # poczatek menu
{

until [ $KONIEC = k ];do
echo "Wciśnij Enter aby kontynuować";
read;
clear; # czyść ekran
echo $KONIEC;
echo " Proste Menu ";
echo;
echo "s - wyświetl ścieżkę w której znajduje się skrypt.";
echo "p - wyświetl listę procesów użytkownika.";
echo "w - wyświetl wszystkie pliki w tym katalogu.";
echo "k - zakończ skrypt.";
read KONIEC;
wybor;
done;
}
menu;
```

Dzięki komendzie read możemy wpisywać z klawiatury dane do zmiennych.

1. W menu pętla until sprawdza czy jest spełniony warunek..
2. Zaczyna się funkcja, wyświetla menu i następnie "wchodzi" do funkcji wybor
3. W funkcji wybor instrukcja case wybiera naszą opcje i wywołuje ją.
4. Powrót do menu

```
#!/bin/bash
INTERFACE=ppp0
NETMASK=192.168.1.0/24
/sbin/ipchains -x
/sbin/ipchains -F
```

/sbin/ipchains -P forward DENY
/sbin/ipchains -A forward -i $INTERFACE -j MASQ -s $NETMASK
echo 1 > /proc/sys/net/ipv4/ip_forward

Skrypt ten tworzy dwie zmienne **INTERFACE** i **NETMASK** w pierwszej zapisana jest nazwa interfejsu sieciowego w drugiej maska sieci(w przypadku zmiany rodzaju połączenia znacznie ułatwi to nam pracę). Kolejne dwa polecenia oraz czyszczą ewentualne reguły firewala ipchains kontrolującego przepływ pakietów. Trzecie polecenie ustala, że pakiety domyślne nie będą przekazywane. Czwarte polecenie określa, że pakiety z sieci lokalnej przeznaczone dla INTERFACE czyli naszego ppp0 będą maskowane. Ostatnie polecenie włącza obsługę przekazywania pakietów TCP/IP w jądrze systemu.

#!/bin/bash
 # zamontowanie CD-ROM
mount -t iso9660 -o ro /dev/hdc /mnt/cdrom
#wpisanie do pliku liczbabl ilosci blokow
df | grep "/mnt/cdrom" | cut -c31-36 > liczbabl
#podanie wyniku na ekran
echo "Liczba blokow na CD-ROM-ie wynosci: " 'cat liczbabl'
#skasowanie niepotrzebnego już pliku
rm liczbapl
umount /mnt/cdrom

W skrypcie tym na początku montujemy cd-rom. Następnie poprzez łączenie potoków (ang.pipes) , z rezultatu polecenia **df** wybierany jest wiersz dotyczący CD-ROM-u a następnie z tego wiersza wycina (cut) łańcuch sześciu kolejnych znaków (-c31-36) od znaki 31 do 36 włącznie. Ten łańcuch znaków to sześciocyfrowa liczba bloków.(po 1024kb czyli w istocie pojemność cd-romu) Łańcuch jest wpisywany do świeżo tworzonego pliku liczbabl. W trzecim wierszu wyprowadzamy rezultat na ekran. Służy do tego celu polecenie echo. Następnie kasujemy niepotrzebny już nam plik i odmontowujemy cd-rom.

Zastosowania skryptów BASH

Skrypty BASH są niezwykle przydatną w systemie UNIX rzeczą.
Pozwalają zautomatyzować wiele działań, dzięki czemu oszczędzają pracy administratorowi systemu.
Za ich pomocą można wykonywać często powtarzające się, rutynowe czynności.
Także z punktu widzenia użytkownika, skrypty okazują się być bardzo użyteczne.
Mogą pełnić wiele funkcji: od generowania stron internetowych, statystyk obciążenia sieci, zajętości powierzchni dysku, aż po nadzorowanie skomplikowanych funkcji systemu.
Możliwości skryptów są ograniczone tylko wiedzą i inwencją autora, jego prawami w systemie, a także możliwościami sprzętowymi komputera.