Powłoka bash

Interpretuje i wykonuje polecenia wydawane z klawiatury lub podane w pliku. Oddziela użytkownika od jądra systemu

Opis ogólny

Podstawowym zadaniem powłoki jest wykonywanie poleceń. Ma możliwość łączenia poleceń (strumienie) i przekierowania wejścia i wyjścia. Posiada możliwość organizowania pętli i instrukcji warunkowych.

Cechy

- Posiada listę "zmiennych powłoki" albo "parametrów", które mogą być tworzone i zmieniane przez użytkownika
- Powłoka korzysta z kilku zmiennych, które definiują "środowisko", w którym ją uruchomiono. Są to tzw. „zmienne środowiska"
- Oprócz powyższych zmiennych powłoka zarządza także własnymi parametrami
- Powłoka wykorzystuje znaki uogólniające (? *)
- Powłoka zawiera wiele wewnętrznych poleceń, które są używane w skryptach powłoki.

Polecenia proste i kod wyjścia

Polecenie proste np. ls -la .

Kod wyjścia - jeśli 0 to wszystko przebiegło poprawnie.

Lista poleceń   ; & | || &&

; - wykonywanie synchroniczne (następne polecenie jest uruchamiane po zakończeniu poprzedniego.

& - — " — asynchroniczne

| — potok (przekazanie wyjścia pierwszego polecenia do następnego.

|| , && - wykonuje następne jeśli poprzednie zakończy-ło się poprawnie.

Grupowanie poleceń () oraz { }

( ) - mogą być użyte do grupowania listy (lista)

{ } - inaczej { lista; }

odstępy

Polecenia strukturalne

Podobne jak w językach programowania (for, while, until) — Pętle
(case, if) ← instr. warunk

Pętla while

```
while lista1
do
    lista2
done
```

Przykład (echo $user)

```
while who | grep ewelb >/dev/null
do
  echo Ewelina jest znowu!
  sleep 5
  echo Koniec wiadomości
done &
```

Pętla until

```
until lista1
do lista2
done
```

Przykład 2

```
until who | grep godzilla >/dev/null
do
  echo Godzilla uciekła
  sleep 4
done &
```

Pętla for

```
for zmienna in lista
do
    lista polecen
done
```

Przykład 3

```
for ptak in kruk szpak kawka
do
  echo to jest $ptak
done
```

Jeżeli for napotyka w trybie, to parametry pozycyjne reprezentują argumenty skryptu. Można użyć parametrów pozycyjnych następująco.

for plik in $* lub for plik in $@ lub for plik ← to samo

wtedy zakłada się, że jest $@

If jest instrukcją warunkową

Najprostsza postać to

```
if lista1
then
    lista2
fi
```

lub

```
if lista1
then lista2
else lista3
fi
```

może być też zestaw testów z użyciem elif

```
if lista1
then
    lista2
elif lista3
then
    lista4
else
    lista5
fi
```

Przykład 4

```
if finger ewurb | grep shell >/dev/null
then
  echo Katalog i shell Andrzeja finger ewurb | grep shell
fi
```

Struktura case pozwala dokonać wyboru na podstawie wyrażenia

```
case słowo in
    wzorzec1) lista1;;
    wzorzec2) lista2;;
    :
esac
```

Definiowanie funkcji

```
nazwa() { lista; }
```

np

```
L() { ls -l | more; }
```

```
L() { ls -l $* | more; }
```

można podawać

Po nazwie funkcji parametry oznaczające katalogi.

Przykład 5

```
L /
```

wypisze na ekranie najpierw katalog bieżący a potem główny

Przykład 6

```
L /home
```

wypisze najpierw katalog główny /, a potem domowy

Strategie powłoki

Parametry pozycyjne

$0 – nazwa skryptu

$1 ... $9 – kolejne parametry skryptu

Parametry kluczowe

nazwa=wartość

$nazwa

(zmienne powłoki)

Parametry automatyczne

$# – ilość parametrów pozycyjnych (skierowanych)

$* – zwraca listę podaną do powłoki w chwili jej uruchomienia

$? – wartość ostatnio wykonanego polecenia. 0 albo kod+128

$$ – numer identyfikacyjny danej powłoki

- pid ostatniego uruchomionego w tle procesu.

Wyrażenia z parametrów

$parametr - wartość parametru
${parametr} - wartość parametru

Przykład

```
bestia=tygrys
echo 0to $bestiay
0to
echo 0to ${bestia}y
0to tygrysy
```

Nadawanie zwartej wartości pustej

```
nazwa=
nazwa=''
nazwa=""
```

Interpretacja linii poleceń:

Polecenie echo $SHELL *pwd* staje się przed wykonaniem

poleceniem echo /bin/bash /home/ameb

Apostrofy i cudzysłowy

Niektóre znaki np. * $ mają specjalne znaczenie dla powłoki.
Można je pozbawić znaczenia na trzy sposoby:

* - oznacza * bez specjalnego znaczenia
\ - można użyć przed enterem co pozwala kontynuować dłuższe polecenie w następnej linii.
'' - zawarte pomiędzy apostrofami są bez znaczenia.

```
echo $HOME          echo '$HOME'
/home/ameb          $HOME
```

" cudzysłowy

Przykład

```
planeta=mars
echo '$planeta' "$planeta" "\$planeta"
$planeta mars $planeta
```

Ciąg znaków w apostrofach może zawierać spacje między słowami

```
psy='ator burek'
echo $psy
ator burek
```

Różnica między $* i $@. Jeśli użyjemy cudzysłowiów to
"$*" - oznacza listę parametrów pozycyjnych oddzielonych spacjami "$1$2...."
"$@" - oznacza to samo co "$1" "$2" ...

Obliczenia [redacted] expr

```
expr 1 + 2
3
expr 2 \* 2 + 4 - 5
3
a=6
expr $a +3
```

[redacted]

```
c=`expr 2 + 7`
echo $c
9
x=`expr 5 % 3` (reszta z dzielenia)
echo $x
2
factor 12345
3 5 823
factor 1001
7 11 13
expr 7 \* 11 \* 13
1001
```

Instalacja csh i tcsh

```
sudo apt install tcsh
```

Można używać csh w którym operacje matematyczne za znakiem @

Np.

```
@ b=1 +2
set a=(1 2 3)
@ a[2]=$a[1] + 5
echo $a
1 6 3
```