ДОКЛАД

По Операционни системи

Тема – Shell скриптове

1.Shell скрипт:

- Скрипт на Shell е набор от команди, които трябва да бъдат изпълнени, така че черупката да може да ги изпълни.

- Твърди се, че е комбинацията от дълги и повтарящи се командни последователности в един скрипт, така че да може да се изпълнява както и когато е необходимо.

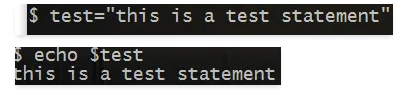
- Основната идея зад създаването на скрипт за черупки е да се намали натоварването на крайния потребител.

2.Основни командни скриптове за Shell:

* Команда Echo : Командата echo се използва за печат.



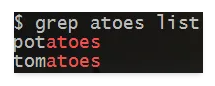
* Използване на променлива: Променлива се използва за съхраняване на някаква стойност, така че цялата стойност не е необходимо да се повтаря и вместо това могат да се използват променливи.



* Cat: Тази команда Shell Scripting се използва за показване на съдържанието на файла. Това е една от много важните команди, използвани в Linux.



* Grep: Друга много важна команда, която се използва за търсене на определен текст във файла и генериране на изхода за вас, свързан със съответстващия модел.



* По-голям от символа (>): Използва се, ако искате да запишете написания на конзолата текст, който да бъде записан във файла.



И други;

3. Създаване на променливи на Shell:

Започва се с дефиниране на променлива на shell в рамките на текущата ни сесия. Това е лесно за постигане,трябва само да посочи име и стойност.Придържаме към конвенцията за запазване на всички главни букви за името на променливата и ще го настроим на обикновен низ.

TEST\_VAR='Hello World!'

Тук сме използват кавички, тъй като стойността на нашата променлива съдържа интервал. Освен това се използват единични кавички, защото удивителният знак е специален знак в bash shell, който обикновено се разширява до bash хронологията, ако не е екраниран или поставен в единични кавички.

Вече имаме променлива на shell. Тази променлива е налична в текущата ни сесия, но няма да бъде предадена на дъщерни процеси.

Може да видим това чрез grepping за нашата нова променлива в изхода set:

set | grep TEST\_VAR

Output

TEST\_VAR='Hello World!'

Може да проверим, че това не е променлива на средата, като опитаме същото с printenv:

printenv | grep TEST\_VAR

Не трябва да се връща изход.

Нека приемем това като възможност да демонстрираме начин за достъп до стойността на всяка обвивка или променлива на околната среда.

echo $TEST\_VAR

Output

Hello World!

Както можеда се види, препратете към стойността на променлива, като поставите пред нея знак $. Shell приема това, за да означава, че трябва да замести стойността на променливата, когато попадне на това.

Така че сега имаме променлива на обвивката. Не трябва да се предава на никакви дъщерни процеси. Можем да създадем нова bash обвивка от нашата текуща, за да демонстрираме:

bash

echo $TEST\_VAR

Ако напишем bash, за да създадем дъщерна обвивка, и след това се опитаме да получим достъп до съдържанието на променливата, нищо няма да бъде върнато.

Върнете се към нашата оригинална обвивка, като напишете exit:

Exit

4.Условни оператори в shell скриптирането:

Една от най-важните части на условното програмиране са изразите if-else. Оператор if-else ви позволява да изпълнявате итеративни условни оператори във вашия код.

Използваме if-else в шел скриптове, когато искаме да оценим дадено условие, след което решаваме да изпълним един набор между два или повече набора оператори, използвайки резултата.

Това по същество ни позволява да изберем отговор на резултата, който оценява нашият условен израз.

Как работи if-else в шел скриптове?

Нека да разгледаме синтаксиса на блока условие if-else.

if [condition]

then

statement1

else

statement2

fi

Тук имаме четири ключови думи, а именно if, then, else и fi.

Ключовата дума if е последвана от условие.

Това условие се оценява, за да се реши кой оператор ще бъде изпълнен от процесора.

Ако условието се изчисли като TRUE, процесорът ще изпълни оператора(ите), последван от ключовата дума then. В синтаксиса се споменава като statement1.

В случай, когато условието се изчисли като FALSE, процесорът ще изпълни оператора(ите), последван от ключовата дума else. Това се обозначава като statement2 в синтаксиса на функцията.

5.Цикли в shell скриптирането:

- Можете да използвате различни цикли в зависимост от ситуацията като например:

# 1) Изложение на цикъл Unix за

Пример: Тази програма ще добави 1 + 2 + 3 + 4 + 5 и резултатът ще бъде 15

for i in 1 2 3 4 5 do sum=`expr $sum + $i` done echo $sum

# 2) Изявление на цикъл Unix While

Пример: Тази програма ще отпечата стойността на „а“ пет пъти, от 1 до 5.

a=1 while [ $a -le 5 ] do echo “value of a=” $a a=`expr $a + 1` done

# 3) Изявление на цикъл Unix Until

Тази програма ще отпечата стойността на „а“ два пъти от 1 до 2.

a=1 until [ $a -ge 3 ] do echo “value of a=” $a a=`expr $a + 1` done

Докато изпълнявате тези цикли, може да се наложи да излезете от цикъла в някакво състояние, преди да завършите всички итерации или да рестартирате цикъла, преди да завършите останалите изрази. Това може да се постигне с изявленията ‘break’ и ‘continue’.

6.Създаване на собствени функций в Shell:

<https://bg.if-koubou.com/articles/how-to/create-your-own-quick-calculator-function-for-the-terminal.html>

Източници:

<https://bg.education-wiki.com/2242610-shell-scripting-commands>

<https://bg.linux-console.net/?p=4048#gsc.tab=0>

<https://bg.myservername.com/dragon-ball-the-breakers-zaklucva-datata-na-startirane-pokazva-frieza>

Изготвил:Валентин Кьосев , 11а клас