Последние новости:

ФЕВ 19	ФЕВ 19	ФЕВ 17	ФЕВ 17	ФЕВ 16	ФЕВ 16
AMD Radeon RX…	Утечка процес…	Pimp My Ride …	Лучший способ…	LIAN LI созда…	Пользователи …

Поиск по сайту:

Поиск по сайту Найти

Идеи становятся силой, когда они овладевают массами (В.И. Ленин).

Использование оператора =~ в Bash

1 мин для чтения

Добавить в избранное

Главное меню » Linux » Использование оператора =~ в Bash

24.05.2023

Регулярное выражение — очень полезный инструмент для сопоставления любого содержимого или поиска и замены содержимого файла или строки с использованием шаблона регулярного выражения. Его можно использовать со сценарием Bash по-разному. Символ =~ используется в операторе if в Bash для поиска любой строки. Многие типы выражений могут использоваться для определения соответствующих шаблонов регулярных выражений. В этой статье объясняются некоторые часто используемые регулярные выражения и использование некоторых выражений с оператором =~.

Часто используемые регулярные выражения

Выражение	Цель
.	Он используется для поиска символов без новой строки (\n).
^	Используется для поиска символов в начале строки.
$	Используется для поиска символов в конце строки.
[0-9]	Он используется для поиска любого числа в диапазоне от 0 до 9 в строке.
[A-Z]	Используется для поиска любого символа из диапазона AZ в строке.
[a-z]	Он используется для поиска любого символа и числа из диапазона az в строке.
[^A-Z0-9]	Он используется для поиска всех символов, кроме заглавных букв и цифр в строке.
[a-zA-z0-9]	Он используется для поиска любого символа и числа из диапазона az, AZ и 0-9 в строке.
\n	Он используется для поиска символа новой строки.
\t	Он используется для поиска символа табуляции.

Различные примеры операторов =~

В этой части статьи показаны различные способы поиска конкретной строки в тексте с использованием оператора =~ и шаблона регулярного выражения.

Пример 1. Поиск определенной строки с использованием символа «*»

Создайте файл Bash со следующим сценарием, который принимает значение основной строки, в котором выполняется поиск строки, и значение строки поиска ищется в значении основной строки. Затем оператор «=~» используется со строкой поиска, чтобы проверить, существует ли строка поиска в основной строке или нет. Здесь символ «*» используется для обозначения любого количества символов.

- Лестницы на металлическом каркасе: надежность, прочность, универсальность
- Что такое высокоуровневый дизайн
- Будущее виртуальных выделенных серверов: трансформация и эволюция
- AMD Radeon RX 7900 GRE сложно продать по сравнению с Nvidia RTX 4070
- Популярные игры на Android за 2024 год: путешествие в мир новых возможностей
- Утечка процессора Intel Lunar Lake показывает что-то странное с кэшами

ПАРТНЕРКА

ОПРОСЫ

Какую ОС на основе Linux вы используете?

- Ubuntu
- CentOS
- Debian
- openSUSE
- Fedora
- Arch Linux
- Другую

Ответ

Посмотреть результаты

- Архив опросов

СВЕЖИЕ КОММЕНТАРИИ

- Аноним к записи Сталкер 2: Сердце Чернобыля, включение NFT
- Аноним к записи Лицензия Минкультуры: Путеводитель в мир сохранения культурного наследия
- Аноним к записи Razer хочет, чтобы вы подключили подушку стула к компьютеру
- Аноним к записи Debian GNU/Linux: 5 Удивительных Фактов и Мелочей
- Аноним к записи Разрешения в Linux

Игровые новости

```
#!/bin/bash

#Возьмем основную строку

read -p "Введите основное строковое значение: " strValue

#Введите строку поиска

read -p "Введите значение в строке поиска: " search

#Проверьте, существует ли строка поиска в основной строке или нет

if [[ $strValue =~ .*$search.* ]]; then

echo "Строка существует в тексте."

else

echo "Строка не существует в тексте."

fi
```

Следующий вывод появляется после выполнения скрипта со значением основной строки «Learn регулярное выражение» и значением строки поиска «regular». Здесь строка поиска существует в основной строке:

```
andreyex@andreyex:-$ bash regex1.bash
Введите основное строковое значение: Learn regular expression
Введите значение в строке поиска: regular
Строка существует в тексте. andreyex@andreyex: -$
```

Пример 2: проверьте расширение конкретного файла

Создайте файл Bash со следующим сценарием, который берет имя файла из аргумента командной строки и проверяет, является ли файл файлом Bash или нет.

```
#!/bin/bash

#Возьмите имя файла из аргумента

filename=$1

#Определите значение расширения для поиска

extension='bash'

#Проверьте, совпадает ли расширение с расширением файла или нет

if [[ "$filename" =~ \.$extension$ ]]; then

echo "$filename это файл bash."

else

echo "$filename это не файл bash."

fi
```

Читать Как включить и отключить Wayland в Ubuntu

Следующий вывод появляется для имени файла «ping1.bash», который является файлом Bash:

```
andreyex@andreyex:-$ bash regex2.bash ping1.bash
ping1.bash это файл bash.
andreyex@andreyex:~$
```

Следующий вывод появляется для имени файла «hello.txt», которое не является файлом Bash:

```
andreyex@andreyex:~$ bash regex2.bash hello.txt
hello.txt это не файл bash.
andreyex@andreyex:-$
```

Пример 3. Поиск определенных символов в строке

Создайте файл Bash со следующим сценарием, который принимает строковое значение и выполняет поиск диапазона символов от «a» до «e» в строке.

```
#!/bin/bash

#Возьмем основную строку

read -p "Введите основное строковое значение: " strValue

#Проверьте, содержит ли строка какой-либо символ от a до e или нет

if [[ $strValue =~ [a-e] ]]; then

echo "Строка содержит символы от 'a' до 'e'"

else

echo "Строка не содержит ни одного символа от "a" до "e""

fi
```

После выполнения скрипта с входным значением «Destroyer» появляется следующий вывод:

```
andreyex@andreyex:~$ bash regex3.bash
Введите основное строковое значение: Destroyer
Строка не содержит ни одного символа от "a" до "e"
andreyex@andreyex: ~$
```

После выполнения скрипта с входным значением «Hello World» появляется следующий вывод:

```
andreyex@andreyex:-$ bash regex3.bash
Введите основное строковое значение: Hello World
Строка содержит символы от 'a' до 'e'
andreyex@andreyex: $
```

Пример 4: Проверка номера мобильного телефона

Создайте файл Bash со следующим сценарием, который берет номер мобильного телефона определенного формата и проверяет, является ли номер действительным или недействительным, используя шаблон регулярного выражения и оператор =~.

```
#Введите номер мобильного телефона в заданном формате

read -p "Введите номер мобильного телефона [880-XXXX-XXXXXX]: " mobile

#Установите шаблон для сопоставления

regexPattern='^880-[0-9]{4}-[0-9]{6}'

#Проверьте, действителен ли номер мобильного телефона или нет

if [[ $mobile =~ $regexPattern.* ]]; then

echo "Номер мобильного телефона действителен."

else

echo "Номер мобильного телефона недействителен."

fi
```

Следующий вывод появляется после выполнения сценария с допустимым входным значением «880-2222-033370»:

```
andreyex@andreyex:~$ bash regex4.bash
Введите номер мобильного телефона [880-XXXX-XXXXXX]: 880-2222 -033370
Номер мобильного телефона действителен.
andreyex@andreyex: ~$
```

Читать Алгоритм Крускала

Следующий вывод появляется после выполнения сценария с недопустимым входным значением «880-12345-67890»:

```
andreyex@andreyex:-$ bash regex4.bash
Введите номер мобильного телефона [880-XXXX-XXXXXX]: 880- 12345-67890
Номер мобильного телефона недействителен.
andreyex@andreyex:~$
```

Заключение

В этой статье показаны методы использования оператора «=~» для поиска строковых значений с различными типами регулярных выражений.

Если вы нашли ошибку, пожалуйста, выделите фрагмент текста и нажмите **Ctrl+Enter**.

Просмотров поста: 120

(**1** оценок, среднее: **5,00** из 5)

Поделиться в соц. сетях:

Тэги: , Bash, ядро Linux **Категории:** Linux

0

Рейтинг статьи

Подписаться авторизуйтесь

Оставьте первый комментарий!

{} [+]

**ссылки nofollow

0 КОММЕНТАРИЕВ

Это может быть вам интересно

4 июля, 2022 4 ноября, 2022 11 мая, 2021 15 августа, 2023

Команда ng не найдена Как установить iptabl… Управление пакетами L… Как освободить место …