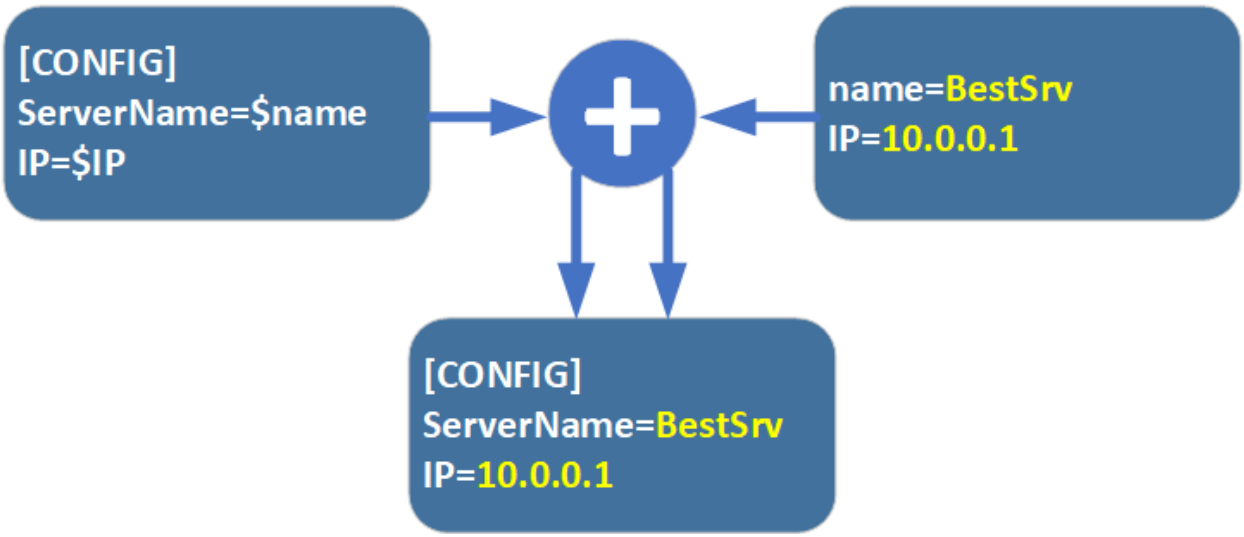
6 вариантов генерации конфигурационных файлов Shell-скриптами



Подготовка конфигурационных файлов из шаблонов — весьма распространённая задача системного администрирования. Решать её можно разными способами, каждый из которых хорош по-своему, здесь же мы рассмотрим, как это сделать с помощью Shell-скриптов.

Учебная задача

Наше исследование проведём в виде решения типовой задачи, в которой необходимо создать Shell-скрипт, генерирующий конфигурационный файл с настройками сетевого стека узла. Формат конфигурационного файла соответствует /etc/network/interfaces. Примем, что настраиваемый узел обладает только одним сетевым интерфейсом с именем «ens33». Скрипт должен обеспечивать возможность указания статического IP-адреса, маски подсети и шлюза по умолчанию.

С учётом вышесказанного получаем следующий шаблон конфигурационного файла:

```
auto ens33
iface ens33 inet static
address ЗНАЧЕНИЕ_IP_АДРЕСА
netmask ЗНАЧЕНИЕ_МАСКИ_СЕТИ
gateway ЗНАЧЕНИЕ_ШЛЮЗА_ПО_УМОЛЧАНИЮ
```

Для дальнейших примеров договоримся, что IP адрес установим в 192.168.0.10, маску подсети 255.255.255.0, а шлюз по умолчанию 192.168.0.1.

Вариант 1. Размещение шаблона внутри скрипта, формирование вывода стандартными средствами

Самый простой вариант решения задачи, которым обычно пользуются начинающие разработчики, — это поместить шаблон в тело скрипта и использовать локальные переменные в качестве подстановочных значений.

```
#!/bin/bash

IP="192.168.0.10"
NETMASK="255.255.255.0"
GATEWAY="192.168.0.1"

TEMPLATE="$(
cat << EOF
auto ens33
iface ens33 inet static
address $IP
netmask $NETMASK
gateway $GATEWAY
EOF
)"

echo -e "$TEMPLATE"
```

Результат работы скрипта (как и всех следующих) будет выглядеть так:

```
auto ens33
iface ens33 inet static
address 192.168.0.10
netmask 255.255.255.0
gateway 192.168.0.1
```

Данный вариант решения всем хорош, за исключением того, что человеческие ошибки (опечатки и прочие), которые могут возникнуть при внесении изменений в шаблон, могут «сломать» скрипт. Поэтому архитектурно более правильным решением будет разнести скрипт и шаблон по разным файлам.

Внимательный читатель может заметить, что мы устанавливаем значения переменных в теле скрипта, и что для генерации следующего конфигурационного файла, с другими настройками всё равно потребуется вносить изменения в скрипт. Да, это так, но сделано это намеренно, для упрощения. Никто же не мешает нам усложнить скрипт и, скажем, реализовать в нём пользовательский интерфейс ввода значений переменных с консоли.

Вариант 2. Размещение шаблона во внешнем файле и его заполнение с помощью envsubst

Создадим файл шаблон «template.txt» следующего содержания:

```
auto ens33
iface ens33 inet static
address $IP
netmask $NETMASK
gateway $GATEWAY
```

Для заполнения шаблона воспользуемся утилитой `envsubst`, которая ищет во входящем потоке строки
вида `$имя_переменной` или `${имя_переменной}` и заменяет их на значение соответствующих переменных. Важно отметить, что для работы этой утилиты переменные должны быть отмечены ключевым словом `export`. С учётом всего вышесказанного наш скрипт будет выглядеть следующим образом:

```
#!/bin/bash

export IP="192.168.0.10"
export NETMASK="255.255.255.0"
export GATEWAY="192.168.0.1"

envsubst  < template.txt
```

Вариант 3. Размещение шаблона во внешнем файле и его заполнение с помощью утилиты sed

Утилита sed позволяет проводить замену одних строк на другие. Чтобы не переделывать шаблон, мы будем искать строки, соответствующие названиям переменных ($IP, $NETMASK, $GATEWAY) и заменять их на их значения. Обратите внимание, что для поиска строки «$IP» мы экранируем служебный символ «$» и ищем строку «\$IP» (аналогично и для других переменных).

```
#!/bin/bash

IP="192.168.0.10"
NETMASK="255.255.255.0"
GATEWAY="192.168.0.1"

cat template.txt | sed "s/\$IP/$IP/" |
                sed "s/\$NETMASK/$NETMASK/" |
                sed "s/\$GATEWAY/$GATEWAY/"
```

Вариант 4. Размещение шаблона во внешнем файле и его заполнение с помощью awk

Данный вариант идейно похож на предыдущий, за исключением того, что вместо sed мы используем awk, что немного сказывается на листинге скрипта.

```
#!/bin/bash

IP="192.168.0.10"
NETMASK="255.255.255.0"
GATEWAY="192.168.0.1"

cat template.txt |
 awk -v replace_str="$IP"      '{ gsub( /\$IP/ ,      replace_str ); print }' |
 awk -v replace_str="$NETMASK" '{ gsub( /\$NETMASK/ , replace_str ); print }' |
 awk -v replace_str="$GATEWAY" '{ gsub( /\$GATEWAY/ , replace_str ); print }'
```

Вариант 5. Размещение шаблона во внешнем файле и его заполнение с помощью eval

Данный вариант отличается от предыдущих тем, что для его реализации не используются никакие внешние утилиты, а лишь встроенная команда `eval`. Скрипт, использующий `eval` для формирования шаблона, будет выглядеть следующим образом:

```
#!/bin/bash

IP="192.168.0.10"
NETMASK="255.255.255.0"
GATEWAY="192.168.0.1"

TEMPLATE=$(< template.txt)

eval "echo -e \"$TEMPLATE\""
```

Вам наверно интересно, зачем вообще нужен `eval`? Понять это можно на примере:

```
#!/bin/bash

VAR="A"

TEMPLATE_LOCAL="VAR=$VAR"
TEMPLATE_FILE="$( < template_file.txt)"
# Содержимое файла template_file.txt:
# VAR=$VAR

echo -e "$TEMPLATE_LOCAL"
# Результат:
# VAR=A
echo -e "$TEMPLATE_FILE"
# Результат:
# VAR=$VAR
eval "echo -e \"$TEMPLATE_FILE\""
# Результат:
# VAR=A
```

Как видите, `eval` позволяет преодолеть особенности Shell, которые заставляют скрипт по-разному интерпретировать одни и те же данные в зависимости от того, заданы ли они в теле скрипта, или же загружены из внешнего файла. Теперь поговорим об особенностях и возможностях, которые даёт использование `eval`.

Первое, на что нужно обратить внимание, так это на необходимость экранирования специальных символов в файле шаблоне. Например, « " » должен быть заменён на « \" ».

Второй особенностью использования `eval`, является возможность писать код в шаблоне. Это позволять строить сложные шаблоны, в которых не только производится подстановка значений, но и могут выводиться (или не выводиться) целые секции в зависимости от заданных условий.

Для демонстрации этой возможности усложним нашу учебную задачу, добавив в неё опциональную настройку второго интерфейса «ens37». То есть «ens33» будет настраиваться всегда, а «ens37» только по необходимости. При использовании `eval` шаблон конфигурационного файла (eval_template.txt) будет выглядеть следующим образом:

```
auto ens33
iface ens33 inet static
address $ENS33_IP
netmask $ENS33_NETMASK
gateway $ENS33_GATEWAY

$(
if [[ (-n ${ENS37_IP}) && (-n ${ENS37_NETMASK}) ]]; then
cat << EOF
auto ens37
iface ens37 inet static
address $ENS37_IP
netmask $ENS37_NETMASK
EOF
fi
)
```

В ранее написанный скрипт внесём небольшие изменения, касающиеся наименования переменных. В остальном всё будет практически без изменений.

```
#!/bin/bash

ENS33_IP="192.168.0.10"
ENS33_NETMASK="255.255.255.0"
ENS33_GATEWAY="192.168.0.1"

ENS37_IP="10.0.0.10"
ENS37_NETMASK="255.255.0.0"

TEMPLATE=$(< eval_template.txt)

eval "echo -e \"$TEMPLATE\""
```

Результат работы скрипта:

```
auto ens33
iface ens33 inet static
address 192.168.0.10
netmask 255.255.255.0
gateway 192.168.0.1

auto ens37
iface ens37 inet static
address 10.0.0.10
netmask 255.255.0.0
```

Кроме возможностей, написание кода в шаблоне несёт и опасность. Дело в том, что враги могут втихаря изменить шаблон, добавив в него вредоносный код, который выполнится в момент генерации очередного конфига. И вместо того, чтобы облегчить себе жизнь автоматизацией рутинных задач, вы получите головную боль, связанную с разгребанием последствий взлома.

В примере выше файл шаблона очень маленький, и его можно быстро просмотреть глазами (при должной квалификации). Но если шаблон большой, то найти вредоносные включения в нём практически нереально. Да и сама идея каждый раз проверять шаблон на предмет несанкционированной модификации — так себе затея.

Вариант 6. Размещение шаблона во внешнем файле и его заполнение с помощью встроенных возможностей Bash

Ещё один вариант чистого Shell, идеологически похожий на `awk` или `sed`. Только здесь мы делаем поиск и замену строк встроенными средствами оболочки.

```
#!/bin/bash

IP="192.168.0.10"
NETMASK="255.255.255.0"
GATEWAY="192.168.0.1"

TEMPLATE="$(< template.txt)"

TEMP="${TEMPLATE/\$NETMASK/$NETMASK}"
TEMP="${TEMP/\$IP/$IP}"
TEMP="${TEMP/\$GATEWAY/$GATEWAY}"

echo -e "$TEMP"
```

Заключение

Мы посмотрели 6 способов построения конфигурационных файлов с помощью Shell-скриптов. Какой из них выбрать? Да тот, что больше нравится!

При прочих равных условиях посмотрите в сторону `envsubst`. Он решает основные задачи и очень компактен в использовании. А вот от использования `eval` с точки зрения безопасности лучше воздержаться.