# **Задание 1**

Дан скрипт:

1. #!/bin/bash
2. PREFIX="${1:-NOT\_SET}"
3. INTERFACE="$2"
5. [[ "$PREFIX" = "NOT\_SET" ]] && { echo "\$PREFIX must be passed as first positional argument"; exit 1; }
6. **if** [[ -z "$INTERFACE" ]]; **then**
7. echo "\$INTERFACE must be passed as second positional argument"
8. exit 1
9. **fi**
11. **for** SUBNET **in** {1..255}
12. **do**
13. **for** HOST **in** {1..255}
14. **do**
15. echo "[\*] IP : ${PREFIX}.${SUBNET}.${HOST}"
16. arping -c 3 -i "$INTERFACE" "${PREFIX}.${SUBNET}.${HOST}" 2> /dev/null
17. **done**
18. **done**

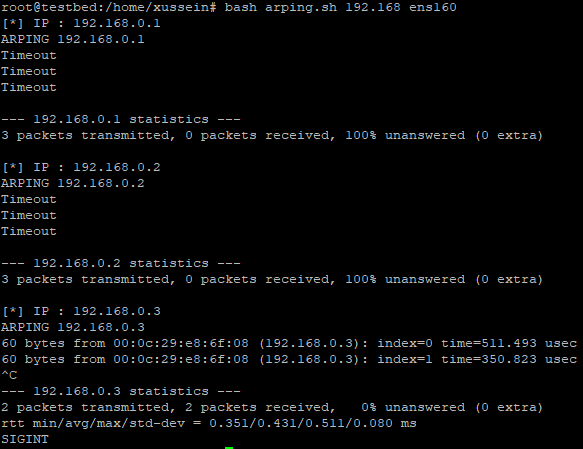
Измените скрипт так, чтобы:

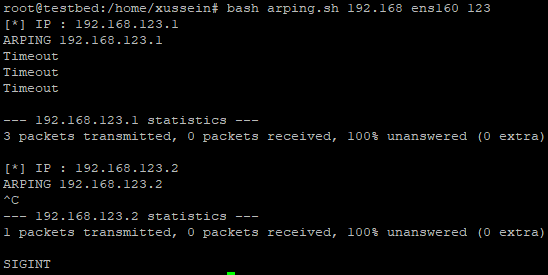
* для ввода пользователем были доступны все параметры. Помимо существующих PREFIX и INTERFACE, сделайте возможность задавать пользователю SUBNET и HOST;
* скрипт должен работать корректно в случае передачи туда только PREFIX и INTERFACE
* скрипт должен сканировать только одну подсеть, если переданы параметры PREFIX, INTERFACE и SUBNET
* скрипт должен сканировать только один IP-адрес, если переданы PREFIX, INTERFACE, SUBNET и HOST
* не забывайте проверять вводимые пользователем параметры с помощью регулярных выражений и знака ~= в условных операторах
* проверьте, что скрипт запускается с повышенными привилегиями и сообщите пользователю, если скрипт запускается без них

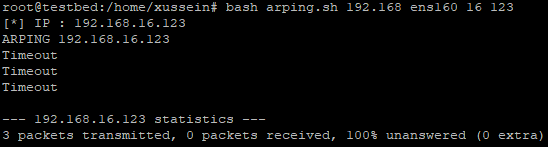
# **Решение 1**

1. #!/bin/bash
2. PREFIX="${1:-NOT\_SET}"
3. INTERFACE="$2"
4. SUBNET="${3:-NOT\_SET}"
5. HOST="${4:-NOT\_SET}"
7. trap "echo 'SIGINT'; exit 1" 2
9. # check if sudo
10. userroot=$(id -nu)
11. **if** [ "$userroot" != "root" ]; **then**
12. echo "Must be root to run script"
13. exit 1
14. **fi**
15. # check if prefix is ok
16. **if** [[ "$PREFIX" = "NOT\_SET" ]]; **then**
17. echo "\$PREFIX must be passed as first positional argument"
18. exit 1
19. **fi**
20. # check if interface is ok
21. **if** [[ -z "$INTERFACE" ]]; **then**
22. echo "\$INTERFACE must be passed as second positional argument"
23. exit 1
24. **fi**
25. # check if subnet is not set and host not set
26. **if** [[ "$SUBNET" = "NOT\_SET" && "$HOST" = "NOT\_SET" ]]; **then**
27. **for** SUBNET **in** {0..255}
28. **do**
29. **for** HOST **in** {1..255}
30. **do**
31. echo "[\*] IP : ${PREFIX}.${SUBNET}.${HOST}"
32. arping -c 3 -i "$INTERFACE" "${PREFIX}.${SUBNET}.${HOST}" 2> /dev/null
33. **done**
34. **done**
35. # check if only subnet is set
36. **elif** [[ ! "$SUBNET" = "NOT\_SET" && "$HOST" = "NOT\_SET" ]]; **then**
37. **if** [[ "$SUBNET" =~ ^([0-9]{1,2}|[0-1][0-9]{0,2}|[2][0-5]{0,2})$ ]]; **then**
38. **for** HOST **in** {1..255}
39. **do**
40. echo "[\*] IP : ${PREFIX}.${SUBNET}.${HOST}"
41. arping -c 3 -i "$INTERFACE" "${PREFIX}.${SUBNET}.${HOST}" 2> /dev/null
42. **done**
43. **else**
44. echo "Subnet is invalid"
45. exit 1
46. **fi**
47. # or all of them are set
48. **else**
49. **if** [[ ! "$SUBNET" =~ ^([0-9]{1,2}|[0-1][0-9]{0,2}|[2][0-5]{0,2})$ || ! "$HOST" =~ ^([0-9]{1,2}|[0-1][0-9]{0,2}|[2][0-5]{0,2})$ ]]; **then**
50. echo "Host and/or subnet is invalid"
51. exit 1
52. **else**
53. echo "[\*] IP : ${PREFIX}.${SUBNET}.${HOST}"
54. arping -c 3 -i "$INTERFACE" "${PREFIX}.${SUBNET}.${HOST}" 2> /dev/null
55. **fi**
56. **fi**









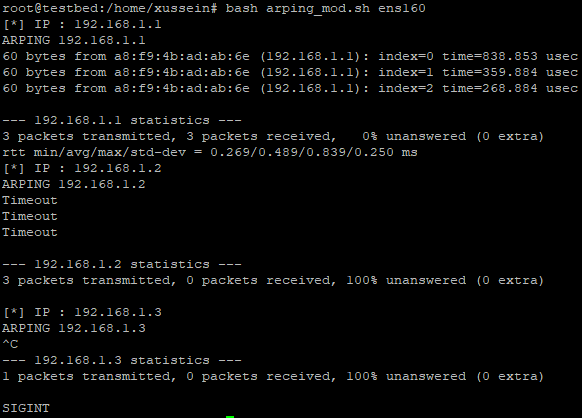
# **Задание 2**

Измените скрипт из Задания 1 так, чтобы:

* единственным параметром для ввода остался сетевой интерфейс;
* определите подсеть и маску с помощью утилиты ip a или ifconfig
* сканируйте с помощью arping адреса только в этой подсети
* не забывайте проверять в начале работы скрипта, что введенный интерфейс существует
* воспользуйтесь shellcheck для улучшения качества своего кода

# **Решение 2**

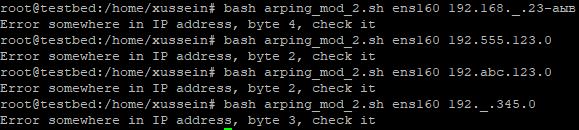
1. #!/bin/bash
2. INTERFACE="$1"
4. trap "echo 'SIGINT'; exit 1" 2
6. **if** [[ -z "$INTERFACE" ]]; **then**
7. echo "You should specify an interface"
8. exit 1
9. **fi**
11. ifconfig $INTERFACE 2> /dev/null > /dev/null
12. **if** [ $? = 0 ]; **then**
13. IFS='.' read -r -a ip\_address <<< $(ip -o -4 addr show ens160 | awk 'NR==1 {print $4}' | sed -E 's/\/[0-9]{1,2}$//')
14. **else**
15. echo "Invalid interface"
16. exit 1
17. **fi**
19. PREFIX=${ip\_address[0]}.${ip\_address[1]}
20. SUBNET=${ip\_address[2]}
22. **for** HOST **in** {1..255}
23. **do**
24. echo "[\*] IP : ${PREFIX}.${SUBNET}.${HOST}"
25. arping -c 3 -i "$INTERFACE" "${PREFIX}.${SUBNET}.${HOST}" 2> /dev/null
26. **done**



Мой вариант скрипта. Логика следующая: если юзер вводит просто интерфейс - проход только по “подсети” интерфейса (eg 192.168.0.1-255). Если юзер вводит дополнительно адрес вида 192.168-192.\_.12, то скрипт будет этот ввод парсить и разбирать, соответственно 168-192 - диапазон, \_ - 0-255, остальное очевидно.

1. #!/bin/bash
2. INTERFACE="$1" # required
3. ADDRESS="$2" # optional, \_.\_.\_.\_ or 10.\_.\_.\_ or 192.168.0.\_ or 183.137.212.11 or even range 123-124.12.12.0-144
5. # trap for SIGINT if something goes south
6. trap "echo 'SIGINT'; exit 1" 2
7. # check if sudo
8. userroot=$(id -nu)
9. **if** [ "$userroot" != "root" ]; **then**
10. echo "Must be root to run script"
11. exit 1
12. **fi**
13. # check if interface string is not empty
14. **if** [[ -z "$INTERFACE" ]]; **then** # if empty
15. echo "You should specify an interface"
16. exit 1
17. **fi**
19. # is address variable is empty we pick interfaces' address and loop through its 'subnet'
20. **if** [[ -z "$ADDRESS" ]]; **then**
21. ifconfig "$INTERFACE" 2> /dev/null > /dev/null
22. res=$?
23. **if** [ $res = 0 ]; **then**
24. IFS='.' read -r -a ip\_address <<< "$(ip -o -4 addr show ens160 | awk 'NR==1 {print $4}' | sed -E 's/\/[0-9]{1,2}$//')"
25. **else**
26. echo "Invalid interface"
27. exit 1
28. **fi**
30. PREFIX=${ip\_address[0]}.${ip\_address[1]}
31. SUBNET=${ip\_address[2]}
33. **for** HOST **in** {1..255}
34. **do**
35. echo "[\*] IP : ${PREFIX}.${SUBNET}.${HOST}"
36. arping -c 3 -i "$INTERFACE" "${PREFIX}.${SUBNET}.${HOST}" 2> /dev/null
37. **done**
38. # if address is not empty we check which bytes are not empty and go through this address, because user knows whats best for him
39. **else**
40. IFS='.' read -r -a ip\_address <<< "$ADDRESS"
41. ranges=(0 0 0 0)
42. **for** ((i=0; i<4; i++)); **do**
43. # if byte is of xxx type
44. **if** [[ ${ip\_address[i]} =~ ^([0-9]{1,2}|[0-1][0-9]{0,2}|[2][0-5]{0,2})$ ]]; **then**
45. ranges[i]=$(eval echo "{${ip\_address[i]}..${ip\_address[i]}}")
46. # if byte is of xxx-xxx type
47. **elif** [[ ${ip\_address[i]} =~ ^([0-9]{1,2}|[0-1][0-9]{0,2}|[2][0-5]{0,2})-([0-9]{1,2}|[0-1][0-9]{0,2}|[2][0-5]{0,2})$ ]]; **then**
48. IFS='-' read -r begin end <<< "${ip\_address[i]}"
49. **if** [ ! "$begin" \> "$end" ]; **then**
50. ranges[i]=$(eval echo "{$begin..$end}")
51. **else**
52. echo "Error in IP address, invalid range"
53. exit 1
54. **fi**
55. # if byte is of \_ type, meaning 0-255
56. **elif** [[ ${ip\_address[i]} = \_ ]]; **then**
57. ranges[i]=$(eval echo "{0..255}")
58. **else**
59. echo "Error somewhere in IP address, byte "$((i+1))", check it"
60. exit 1
61. **fi**
62. **done**
64. # main loop
65. **for** byte1 **in** ${ranges[0]}; **do**
66. **for** byte2 **in** ${ranges[1]}; **do**
67. **for** byte3 **in** ${ranges[2]}; **do**
68. **for** byte4 **in** ${ranges[3]}; **do**
69. echo "[\*] IP : $byte1.$byte2.$byte3.$byte4"
70. arping -c 3 -i "$INTERFACE" "$byte1.$byte2.$byte3.$byte4" 2> /dev/null
71. **done**
72. **done**
73. **done**
74. **done**
75. **fi**

Проверки:



Выполнение:

