2. Изучение сценариев bash

Лысенко Артем, группа 7.1
Исходный код всех скриптов доступен по ссылке:
https://github.com/quicklybly/operating-systems-workshop/tree/master/lab2

1. Печать аргументов с таймаутом (версия с shift/case)

В начале программа проверяет количество переданных параметров, в случае неправильного ввода скрипт печатает инструкцию и завершает свою работу.

```
usage="Usage: ptxt_1 -n <repeat_times> -t <timeout> -- <text>"

if [[ ! $# -eq 6 ]]
then
 echo "Invalid number of arguments"
 echo "$usage"
 exit 2
fi
```

Затем в цикле while происходит считывание параметров командной строки с использованием shift и case.
После этого в цикле while скрипт печатает текст заданное число раз с заданным таймаутом, для создания таймаута используется sleep “$timeout”

Полный код скрипта.

```
#!/bin/bash
```

```
usage="Usage: ptxt_1 -n <repeat_times> -t <timeout> -- <text>"

if [[ ! $# -eq 6 ]]
then
 echo "Invalid number of arguments"
 echo "$usage"
 exit 2
fi

while [[ -n "$1" ]]
do
 case "$1" in
 -n)
  repeat_times="$2"
  shift 2
  ;;
 -t)
  timeout="$2"
  shift 2
  ;;
 --)
  text="$2"
  shift 2
  ;;
 *)
  echo "Invalid argument: $1"
  echo "$usage"
  exit 2
  ;;
 esac
done

while [[ $repeat_times -gt 0 ]]
do
 echo "$text"
 sleep "$timeout"
 ((repeat_times--))
```

```
done
```

2. Печать аргументов с таймаутом (версия с getops)

Аналогично прошлому примеру программа проверяет корректность введенных данных, затем считывает именованные параметры (-n и -t) через getops в цикле while, т.к. Getops не работает с long parameters (--) считывание текста для печати происходит через shift (как в методичке), в конце, аналогично прошлому скрипту, идет цикл while. Для обьявления переменных используется declare, -i - целое число.

Полный код скрипта.

```
#!/bin/bash

usage="Usage: ptxt -n <repeat_times> -t <timeout> -- <text>"

if [[ ! $# -eq 6 ]]
then
 echo "Invalid number of arguments"
 echo "$usage"
 exit 2
fi

declare -i repeat_times
declare timeout
declare text

while getopts "n:t:" option; do
 case "$option" in
   n) repeat_times=$OPTARG ;;
   t) timeout=$OPTARG ;;
   *)
     echo "Invalid argument: $option"
```

```
    echo "$usage"
    exit 2 ;;
 esac
done
shift $((OPTIND - 1))

text=$1

while [[ $repeat_times -gt 0 ]]
do
 echo "$text"
 sleep "$timeout"
 ((repeat_times--))
done
```

3. Сортировка

В качестве алгоритма сортировки используется сортировка пузырьком, оптимизированная для частично отсортированных массивов (естественность сортировки).

В начале скрипта создаем массив из переданных аргументов и обьявляем n - размер массива.

```
declare -a arr=("$@")
declare -i n=$#
```

Если массив пустой, то программа сообщит об этом сообщением “Empty array provided”. После этой проверки программа сортирует массив и печатает результат в консоль командой **echo "${arr[*]}"**

Полный код скрипта

```
#!/bin/bash

declare -a arr=("$@")
declare -i n=$#

if [[ $n -eq 0 ]]; then
 echo "Empty array provided"
 exit 2
fi

for (( i = 0; i < $n; i++ )); do
 swapped=0
 for (( j = 0; j < $n-i-1; j++ )); do
   if [[ ${arr[$j]} > ${arr[$j+1]} ]]; then
     swapped=1
     tmp=${arr[$j]}
     arr[$j]=${arr[$j+1]}
     arr[$j+1]=$tmp
   fi
 done
 if [[ $swapped ]]; then
   break
 fi
done

echo "Result array:"
echo "${arr[*]}"
```