# 12

Рафи Кази ар

НКАбд-03-24

1032238132@pfur.ru

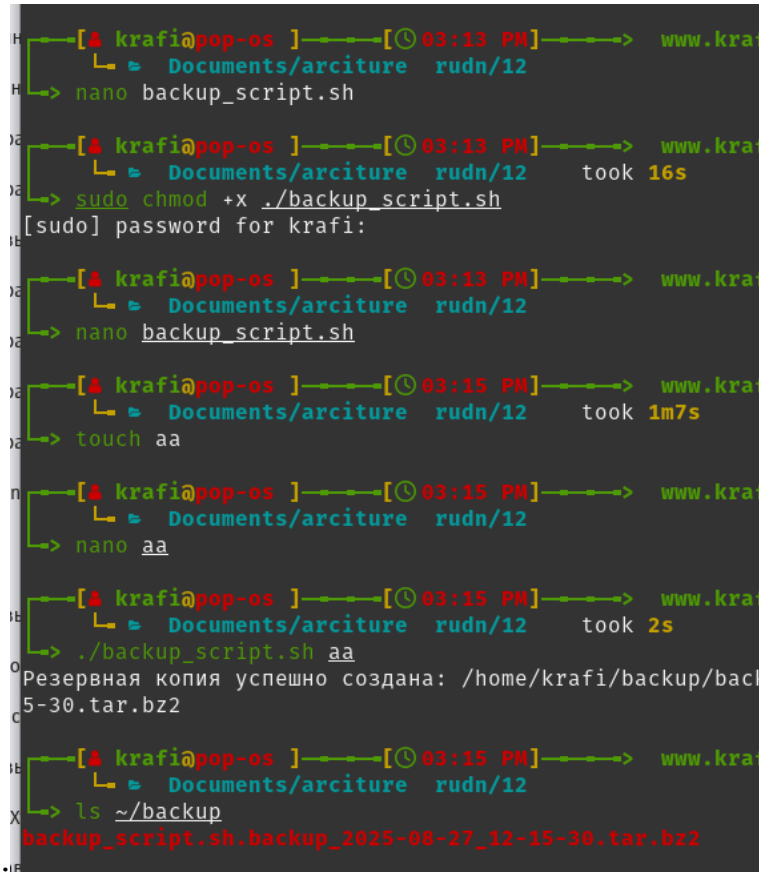
**Задание 1: Скрипт для резервного копирования самого себя**

```
#!/bin/bash

# backup_script.sh
# Скрипт создает резервную копию самого себя в директории ~/backup

# Определяем домашнюю директорию пользователя
backup_dir="$HOME/backup"

# Создаем директорию backup, если её нет
mkdir -p "$backup_dir"

# Имя текущего скрипта
script_name=$(basename "$0")

# Формируем имя для архивной копии с датой
backup_name="${script_name}.backup_$(date +%Y-%m-%d_%H-%M-%S).tar.bz2"

# Архивируем скрипт в tar.bz2 (высокая степень сжатия)
tar -cjf "$backup_dir/$backup_name" "$script_name"

# Проверяем успешность выполнения архивации
```

```
if [ $? -eq 0 ]; then
echo ”Резервная копия успешно создана: $backup_dir/$backup_name”
else
echo ”Ошибка при создании резервной копии!” >&2
exit 1
fi
```

**Пояснение:**

- mkdir -p создает директорию, если она не существует.
- basename ”$0” получает имя самого скрипта.
- date +%Y-%m-%d_%H-%M-%S формирует строку с текущей датой и временем для имени файла.
- tar -cjf создает архив с использованием алгоритма сжатия bzip2.



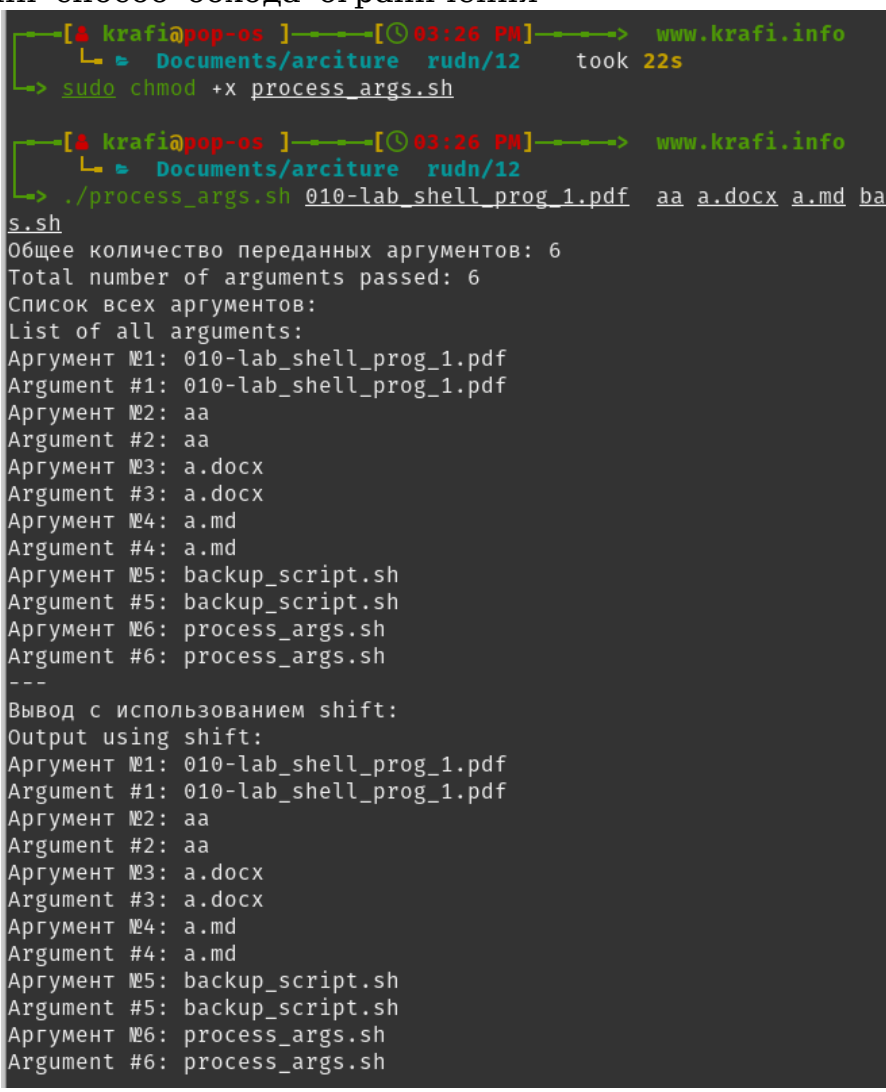
**Задание 2: Скрипт, обрабатывающий произвольное число аргументов**

```
#!/bin/bash
echo ”Общее количество переданных аргументов: $#”
echo ”Total number of arguments passed: $#”
echo ”Список всех аргументов:”
echo ”List of all arguments:”
count=1
for arg in ”$@”; do
echo ”Аргумент №$count: $arg”
echo ”Argument #$count: $arg”
((count++))
done
echo ”---”
echo ”Вывод с использованием shift:”
echo ”Output using shift:”
count=1
while [ $# -gt 0 ]; do
echo ”Аргумент №$count: $1”
echo ”Argument #$count: $1”
((count++))
shift
done
```

**Пояснение:**

- $# содержит количество переданных аргументов.
- ”$@” представляет собой список всех аргументов, каждый в отдельных кавычках, что правильно обрабатывает аргументы с пробелами.

- Цикл while с командой shift обрабатывает аргументы по одному, пока они не закончатся. Это классический способ обхода ограничения на $10, $11 и т.д.




**Задание 3: Упрощенный аналог команды ls**

```
#!/bin/bash

# my_ls.sh
# Упрощенный аналог команды ls.  Выводит информацию о файлах в указанном каталоге.
```

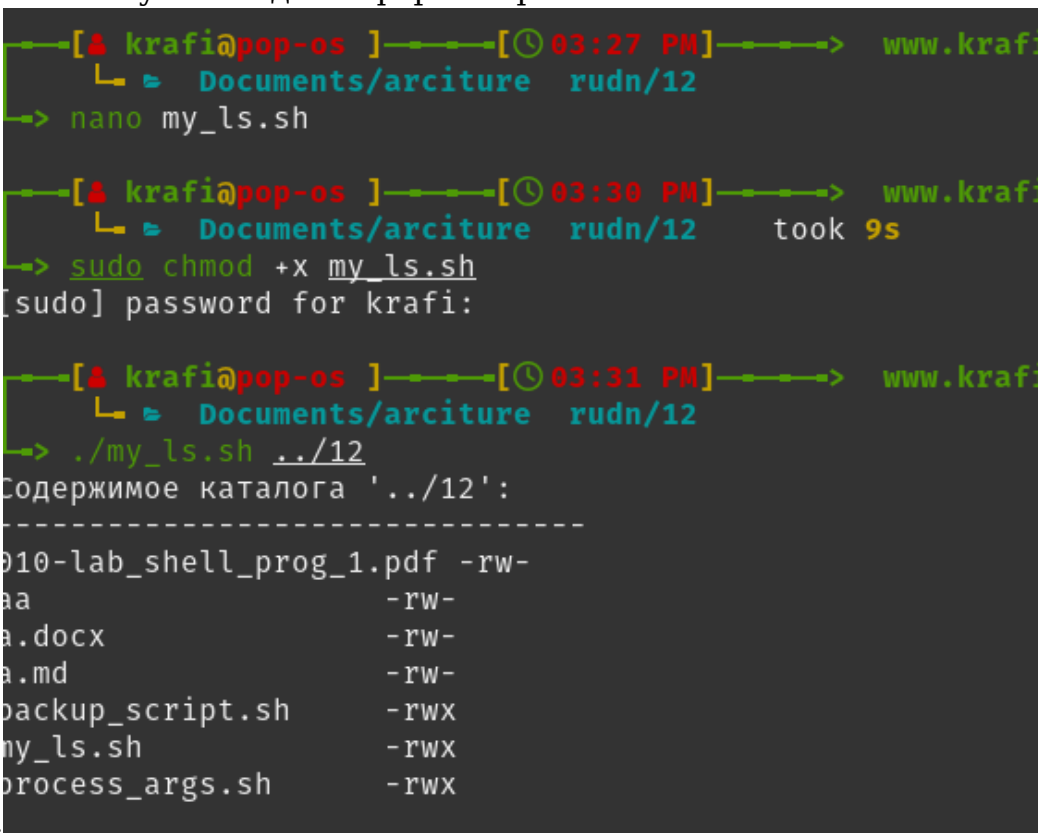
```
# Проверяем, передан ли аргумент (путь к каталогу)
target_dir=”${1:-.}” # Используем текущий каталог (.), если аргумент не передан

# Проверяем, существует ли целевой каталог и является ли он каталогом
if [ ! -d ”$target_dir” ]; then
echo ”Ошибка: ’$target_dir’ не является каталогом или не существует.” >&2
exit 1
fi

echo ”Содержимое каталога ’$target_dir’:”
echo ”--------------------------------”

# Перебираем все элементы в целевом каталоге
for item in ”$target_dir”/*; do
# Получаем только имя файла/каталога без пути
name=$(basename ”$item”)

# Определяем тип и права доступа
if [ -d ”$item” ]; then
type=”d” # directory
elif [ -f ”$item” ]; then
type=”-” # regular file
else
type=”?” # other (link, device, etc.)
fi

# Формируем строку прав доступа
perms=””
[ -r ”$item” ] && perms=”${perms}r” || perms=”${perms}-”
[ -w ”$item” ] && perms=”${perms}w” || perms=”${perms}-”
[ -x ”$item” ] && perms=”${perms}x” || perms=”${perms}-”

# Выводим информацию
printf ”%-20s %s%s\n” ”$name” ”$type” ”$perms”
done
```

**Пояснение:**

- ${1:-.} - параметр подстановки: использует

первый аргумент ($1), а если его нет, то текущую директорию (.).

- [ -d ], [ -f ], [ -r ], [ -w ], [ -x ] - команда test для проверки типа файла и прав доступа.
- basename убирает путь из имени файла.
- printf используется для форматированного вывода.




**Задание 4: Фильтр для поиска файлов по атрибуту доступа**

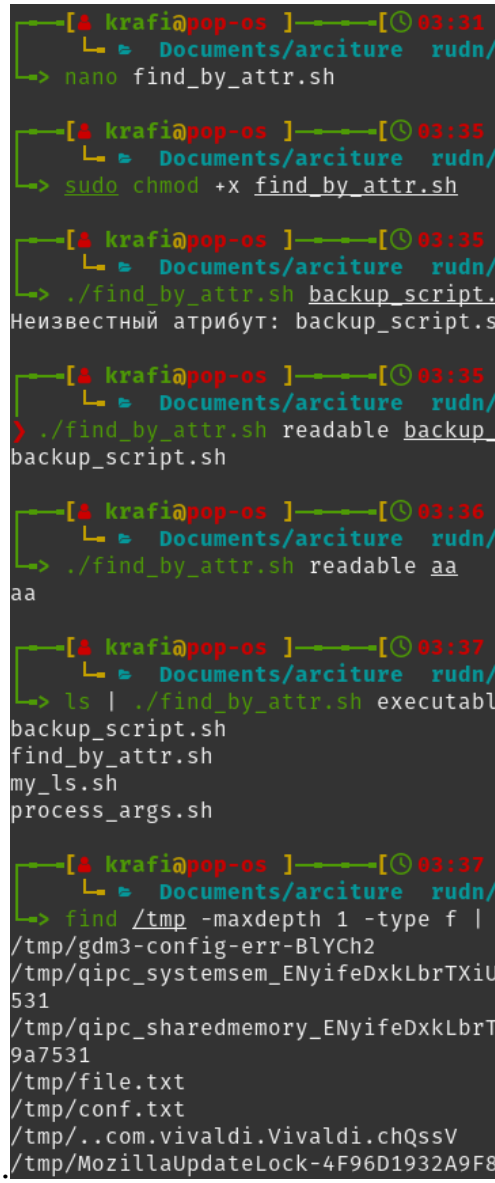
```
#!/bin/bash

# find_by_attr.sh
# Фильтр: читает список файлов из stdin или аргументов и выводит те, что имеют заданный атрибут.
# Использование:
# find_by_attr.sh [атрибут] [файл1 файл2 ...]
```

```
# ls | find_by_attr.sh [атрибут]
# find_by_attr.sh [атрибут] < filelist.txt

# Проверяем, что указан хотя бы один атрибут
if [ $# -lt 1 ]; then
echo ”Использование: $0 {readable|writable|executable} [file ...]” >&2
exit 1
fi

attribute=”$1”
shift # Удаляем первый аргумент (атрибут) из списка, чтобы остались только файлы

# Если файлы не переданы как аргументы, читаем из стандартного ввода
if [ $# -eq 0 ]; then
while read -r file; do
files+=(”$file”)
done
else
files=(”$@”) # Используем оставшиеся аргументы как массив файлов
fi

# Проверяем каждый файл в списке
for file in ”${files[@]}”; do
# Игнорируем пустые строки
if [ -z ”$file” ]; then
continue
fi

# Проверяем атрибут и выводим файл, если условие выполняется
case ”$attribute” in
”readable”)
if [ -r ”$file” ]; then
echo ”$file”
fi
;;
”writable”)
```

```
if [ -w ”$file” ]; then
echo ”$file”
fi
;;
”executable”)
if [ -x ”$file” ]; then
echo ”$file”
fi
;;
*)
echo ”Неизвестный атрибут: $attribute. Используйте: readable, writable, executable.” >&2
exit 1
;;
esac
done
```

**Пояснение:**

- Скрипт может работать как в режиме обработки аргументов, так и в режиме фильтра (чтение из stdin).
- shift удаляет первый обработанный аргумент.
- Цикл while read читает стандартный ввод построчно, если файлы не были переданы явно.
- case используется для выбора действия в




зависимости от запрошенного атрибута.

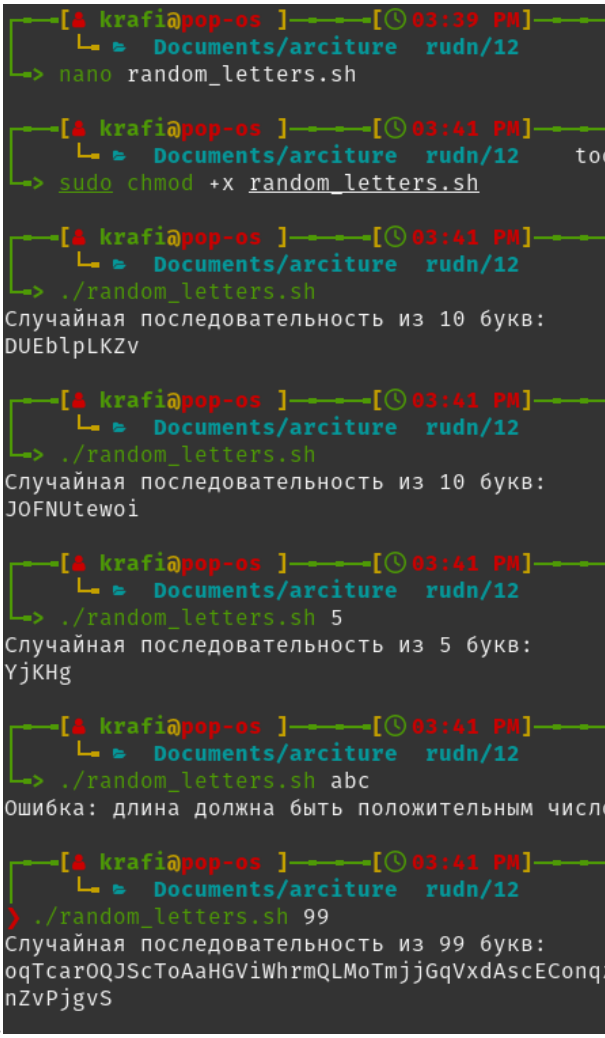
**Задание 5: Генератор случайных последовательностей букв**

```
#!/bin/bash

# random_letters.sh
# Генерирует случайную последовательность букв латинского алфавита.
```

```
# Задаем длину последовательности по умолчанию
length=${1:-10}

# Проверяем, что аргумент - число
if ! [[ "$length" =~ ^[0-9]+$ ]]; then
echo "Ошибка: длина должна быть положительным числом."
>&2
exit 1
fi

# Строка, из которой будем выбирать символы
letters="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"
letters_len=${#letters} # Получаем длину строки letters

echo "Случайная последовательность из $length букв:"
# Генерируем последовательность заданной длины
for (( i=0; i < length; i++ )); do
# Генерируем случайный индекс в диапазоне [0, letters_len -
1]
random_index=$(( RANDOM % letters_len ))
# Извлекаем символ по сгенерированному индексу
printf "%s" "${letters:random_index:1}"
done

# Добавляем перевод строки в конце
echo
```

**Пояснение:**

- ${1:-10} - использует первый аргумент как длину последовательности, по умолчанию 10.
- [[ "$length" =~ ^[0-9]+$ ]] - проверяет с помощью регулярного выражения, что аргумент является числом.
- ${#letters} - возвращает длину строки letters.
- RANDOM % letters_len - генерирует случайное число в диапазоне от 0 до letters_len - 1.
- ${letters:random_index:1} - оператор взятия подстроки: извлекает 1 символ из строки letters,




начиная с позиции random_index.

1.