Лабораторная работа № 13

Программирование в командном процессоре ОС UNIX. Ветвления и циклы.

Павличенко Родион Андреевич

Содержание

1 Цель работы **5**

2 Выполнение лабораторной работы **6**

3 Контрольные вопросы **10**

4 Выводы **12**

Список иллюстраций

2.1 1 задание 6
2.2 2 задание 7
2.3 3 задание 8
2.4 4 задание 9

Список таблиц

1 Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

2 Выполнение лабораторной работы

Используя команды getopts grep, написали командный файл, который анализирует командную строку с ключами

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int number;

    printf("Введите число: ");
    scanf("%d", &number);

    if (number > 0) {
       printf("Число больше нуля\n");
       exit(1);
    } else if (number < 0) {
        printf("Число меньше нуля\n");
       exit(2);
    } else {
        printf("Число равно нулю\n");
        exit(0);
    }
}
```

Рис. 2.1: 1 задание

Написали на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции exit(n), передавая информацию в коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды $?, выдать сообщение о том, какое число было введено.

```
#!/bin/bash

./program

case $? in
    0)
        echo "Число равно нулю";;
    1)
        echo "Число больше нуля";;
    2)
        echo "Число меньше нуля";;
esac
```

Рис. 2.2: 2 задание

Написали командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до (например 1.tmp, 2.tmp, 3.tmp,4.tmp и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют)

```bash
#!/bin/bash

create_files() {
   local count=$1;
   for ((i=1; i<=$count; i++)); do
       touch "$1.tmp"
       echo "Создан файл $i.tmp"
   done
}

delete_files() {
  local count=$1
  for ((i=1; i<=$count; i++)); do
      if [ -e "$i.tmp" ]; then
          rm "$i.tmp"
          echo "Удален файл $i.tmp"
      fi
  done
}

if [ $# -eq 0 ]; then
     echo "Не указано количество файлов для создания"
     exit 1
fi

action=$1

case $action in
     create)
        create_files $2
        ;;
     delete)
        delete_files $2
        ;;
     *)
        echo "Неверное действие"
        exit 1
        ;;
esac
```

Рис. 2.3: 3 задание

Написали командный файл, который с помощью команды tar запаковывает

в архив все файлы в указанной директории. Модифицировали его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find).

```
#!/bin/bash

directory=$1
output_archive="archive.tar.gz"
threshold_days=7

if [ -z "$directory" ]; then
    echo "Укажите директорию в качестве аргумента"
    exit 1
fi

if [ ! -d "$directory" ]; then
   echo "Указанная директория не существует"
   exit 1
fi

find "$directory" -type f -mtime -$threshold_days -print0 | tar --null -czf "$output_archive" --files-from -
echo "Архивация завершена. Архив создан: $output_archive"
```

Рис. 2.4: 4 задание

3 Контрольные вопросы

1. Команда getopts используется для обработки аргументов командной строки в скриптах. Она позволяет анализировать переданные опции (флаги) и их аргументы, упрощая работу с командной строкой.

2. Метасимволы, такие как *, ?, и [], используются для создания списка файлов, соответствующих шаблону. Это называется расширением (глоббингом). Например,* .txt будет соответствовать всем файлам с расширением .txt в текущем каталоге, а file[1-9].txt — файлам с именами, такими как file1.txt, file2.txt и так далее.

3. Операторы управления действиями в Bash включают:

&& — логическое И, выполняет команду только если предыдущая завершилась успешно.

|| — логическое ИЛИ, выполняет команду только если предыдущая завершилас

; — разделяет команды, выполняются последовательно.

() — позволяет выполнить команды в подshell.

{} — позволяет выполнять команды в текущем процессе.

4. Для прерывания цикла в Bash используются:

break — немедленно завершает выполнение цикла.

continue — пропускает текущую итерацию цикла и переходит к следующей.

5. Команды true и false всегда возвращают статус выхода 0 и 1 соответственно. Это полезно для создания условий или управления потоком в скриптах.

Например, false можно использовать для того, чтобы всегда завершать выполнение команды с ошибкой, а true — для успешного завершения.

6. Строка if test -f man*s*/i.$s проверяет, существует ли файл с именем, сформированным из переменных man, $s, $i и $s. В частности:

test -f проверяет, является ли файл обычным файлом.
$i и $s — это переменные, которые подставляются в строку для формирования

7. while выполняет блок команд, пока условие истинно. until выполняет блок команд, пока условие ложно.

4 Выводы

Изучили основы программирования в оболочке ОС UNIX. Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов