Шаблон отчёта по лабораторной работе

Простейший вариант

Дмитрий Сергеевич Кулябов

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

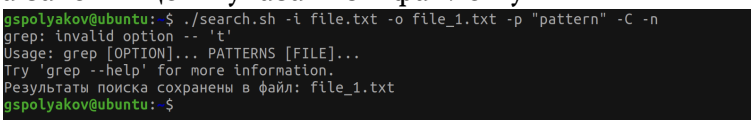
Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

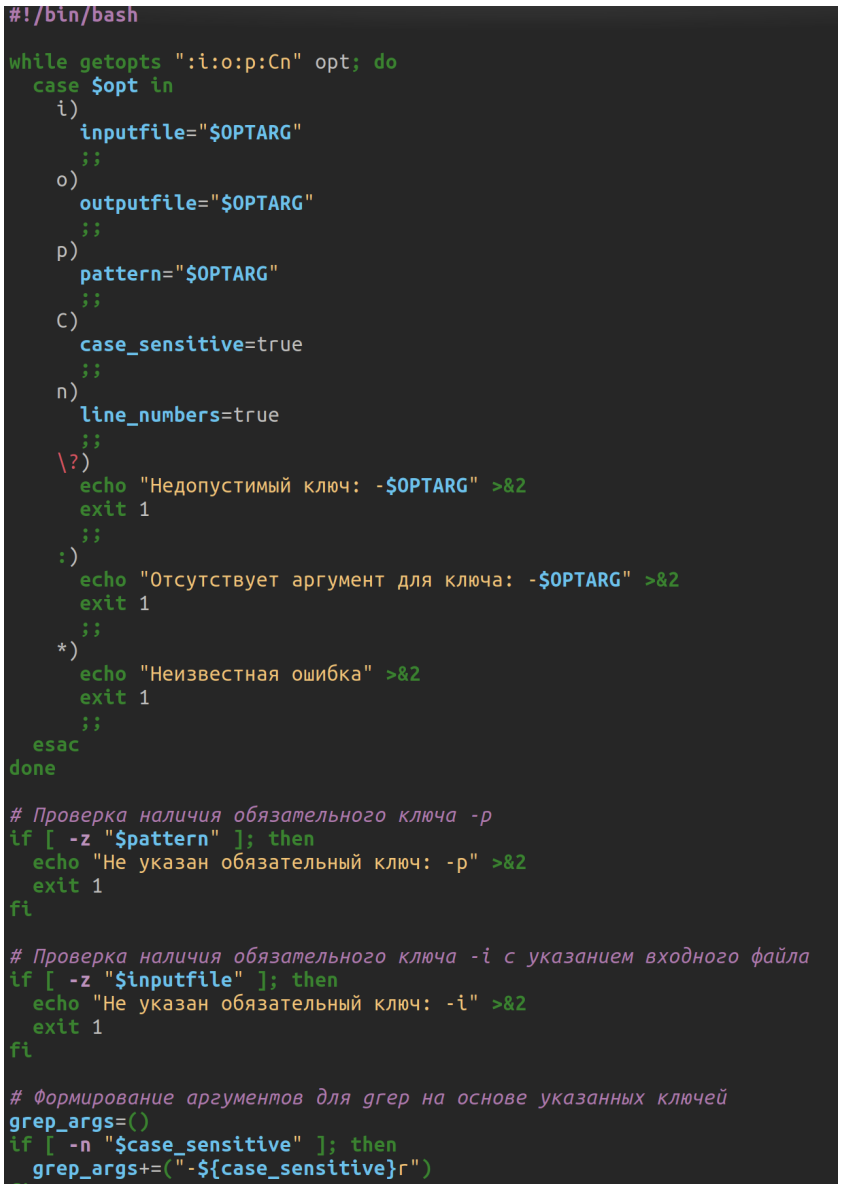
2 Задание

1. Используя команды getopts grep, написать командный файл, который анализирует командную строку с ключами: – -i inputfile — прочитать данные из указанного файла; – -o outputfile — вывести данные в указанный файл; – -p шаблон — указать шаблон дл япоиска; – -C — различать большие и малые буквы; – -n — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом -p.
2. Написать 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 о коде завершения в оболочку. Команд- ный файл должен вызывать эту программу и, проанализировав с помощью команды $?, выдать сообщение о том, какое число было введено.
3. Написать 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).
4. Написать командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find).

3 Выполнение лабораторной работы

1. Используя команды getopts grep, написать командный файл, который анализирует командную строку с ключами, а затем ищет в указанном файле нужные строки, определяемые ключом -p.





```bash
#!/bin/bash

while getopts ":i:o:p:Cn" opt; do
  case $opt in
    i)
      inputfile="$OPTARG"
      ;;
    o)
      outputfile="$OPTARG"
      ;;
    p)
      pattern="$OPTARG"
      ;;
    C)
      case_sensitive=true
      ;;
    n)
      line_numbers=true
      ;;
    \?)
      echo "Недопустимый ключ: -$OPTARG" >&2
      exit 1
      ;;
    :)
      echo "Отсутствует аргумент для ключа: -$OPTARG" >&2
      exit 1
      ;;
    *)
      echo "Неизвестная ошибка" >&2
      exit 1
      ;;
  esac
done

# Проверка наличия обязательного ключа -p
if [ -z "$pattern" ]; then
  echo "Не указан обязательный ключ: -p" >&2
  exit 1
fi

# Проверка наличия обязательного ключа -i с указанием входного файла
if [ -z "$inputfile" ]; then
  echo "Не указан обязательный ключ: -i" >&2
  exit 1
fi

# Формирование аргументов для grep на основе указанных ключей
grep_args=()
if [ -n "$case_sensitive" ]; then
  grep_args+=("-${case_sensitive}r")
fi
if [ -n "$line_numbers" ]; then
```

```bash
# Формирование аргументов для grep
grep_args=()
if [ -n "$case_sensitive" ]; then
  grep_args+=("-${case_sensitive}r"
fi
if [ -n "$line_numbers" ]; then
  grep_args+=("-${line_numbers}n")
fi
grep_args+=("$pattern")
grep_args+=("$inputfile")

# Запуск команды grep с указанными
grep "${grep_args[@]}" > "$outputfi

echo "Результаты поиска сохранены в
```

2. Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю.

```
gspolyakov@ubuntu:~$ ./script_program.sh
Компиляция успешна, выполнение программы...
Введите число: 10
Число больше нуля
gspolyakov@ubuntu:~$
```

```
#!/bin/bash

gcc -o my_program program.c  # Комп

if [ $? -eq 0 ]; then
    echo "Компиляция успешна, выпол
    ./my_program  # Выполнение прог

    case $? in
        0)
            echo "Число равно нулю"
            ;;
        1)
            echo "Число больше нуля
            ;;
        2)
            echo "Число меньше нуля
            ;;
        *)
            echo "Неизвестный код з
            ;;
    esac
else
    echo "Ошибка компиляции програм
fi
```

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

int main() {
  int number;

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

  if (number > 0) {
    exit(1);
  } else if (number < 0) {
    exit(2);
  } else {
    exit(0);
  }
}
```

3. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до ⊠ (например 1.tmp, 2.tmp,

```
gspolyakov@ubuntu:~$ ./script_files.sh -c 5
Создан файл: file1.tmp
Создан файл: file2.tmp
Создан файл: file3.tmp
Создан файл: file4.tmp
Создан файл: file5.tmp
gspolyakov@ubuntu:~$ ./script_files.sh -d
Удален файл: file1.tmp
Удален файл: file2.tmp
Удален файл: file3.tmp
Удален файл: file4.tmp
Удален файл: file5.tmp
gspolyakov@ubuntu:~$
```

3.tmp,4.tmp и т.д.).

```bash
#!/bin/bash

# Проверяем наличие аргументов
if [ $# -eq 0 ]; then
  echo "Ошибка: не указаны параметры командной строки."
  echo "Использование: $0 -c <число_файлов> | -d"
  exit 1
fi

while getopts ":cd" opt; do
  case $opt in
    c)
      create_files=true
      ;;
    d)
      delete_files=true
      ;;
    \?)
      echo "Неверный параметр: -$OPTARG"
      exit 1
      ;;
    :)
      echo "Параметр -$OPTARG требует аргумент."
      exit 1
      ;;
  esac
done

if [ "$create_files" = true ] && [ "$delete_files" = true ]; then
  echo "Ошибка: нельзя указывать одновременно параметры -c и -d."
  exit 1
fi

if [ "$create_files" = true ]; then
  shift $((OPTIND-1)) # Сдвигаем указатель на следующий аргумент пос
  if [ $# -eq 0 ]; then
    echo "Ошибка: не указано число файлов для создания."
    echo "Использование: $0 -c <число_файлов>"
    exit 1
  fi

  num_files=$1 # Получаем число файлов из аргумента командной строки
  prefix="file" # Префикс имени файлов
  suffix=".tmp" # Суффикс имени файлов

  # Создаем файлы
  for ((i=1; i<=$num_files; i++)); do
    file_name="${prefix}${i}${suffix}"
    touch "$file_name"
    echo "Создан файл: $file_name"
  done
elif [ "$delete_files" = true ]; then
  prefix="file" # Префикс имени файлов
  suffix=".tmp" # Суффикс имени файлов

  # Удаляем файлы, если они существуют
  for ((i=1; ; i++)); do
    file_name="${prefix}${i}${suffix}"
    if [ -e "$file_name" ]; then
      rm "$file_name"
      echo "Удален файл: $file_name"
    else
      break
    fi
  done
else
  echo "Ошибка: не указано действие -c или -d."
  echo "Использование: $0 -c <число_файлов> | $0 -
  exit 1
fi

  exit 1
fi
```

4. Написать командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find).

```
gspolyakov@ubuntu:~$ ./script_archive_ver2.sh backup/
tar: backup/backup_20230415_211051.zip: Cannot stat: No such file or directory
tar: Exiting with failure status due to previous errors
Файлы, измененные менее недели назад, были добавлены в архив: archive.tar.gz
gspolyakov@ubuntu:~$
```

```bash
#!/bin/bash

# Проверяем наличие аргументов
if [ $# -ne 1 ]; then
  echo "Ошибка: не указана директория для ар
  echo "Использование: $0 <директория>"
  exit 1
fi

dir_path="$1" # Получаем путь к директории и
archive_name="archive.tar.gz" # Имя архива

# Архивируем все файлы в указанной директори
tar czf "$archive_name" -C "$dir_path" .
echo "Все файлы из директории $dir_path были
```

```
gspolyakov@ubuntu:~$ ./script_archive_ver1.sh backup/
Все файлы из директории backup/ были добавлены в архив: archive.tar.gz
gspolyakov@ubuntu:~$
```

```bash
#!/bin/bash

# Проверяем наличие аргументов
if [ $# -ne 1 ]; then
  echo "Ошибка: не указана директория для архивации."
  echo "Использование: $0 <директория>"
  exit 1
fi

dir_path="$1" # Получаем путь к директории из аргумента командной строки
archive_name="archive.tar.gz" # Имя архива

# Ищем файлы, измененные менее недели назад (7 дней)
files=$(find "$dir_path" -type f -mtime -7)

# Проверяем, найдены ли файлы
if [ -z "$files" ]; then
  echo "Нет файлов, измененных менее недели назад."
else
  # Архивируем найденные файлы
  tar czf "$archive_name" -C "$dir_path" $files
  echo "Файлы, измененные менее недели назад, были добавлены в архив: $archive_name"
fi
```

4 Выводы

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

Список литературы