Front matter

title: "Отчет" subtitle: "Лабораторная работа №11" author: "Подоляк Иван НПМбд-02-21"

Generic otions

lang: ru-RU toc-title: "Содержание"

Bibliography

bibliography: bib/cite.bib csl: pandoc/csl/gost-r-7-0-5-2008-numeric.csl

Pdf output format

toc: true # Table of contents toc-depth: 2 lof: true # List of figures lot: true # List of tables fontsize: 12pt linestretch: 1.5 papersize: a4 documentclass: scrreprt

I18n polyglossia

polyglossia-lang: name: russian options: - spelling=modern - babelshorthands=true polyglossia-otherlangs: name: english

I18n babel

babel-lang: russian babel-otherlangs: english

Fonts

mainfont: PT Serif romanfont: PT Serif sansfont: PT Sans monofont: PT Mono mainfontoptions: Ligatures=TeX romanfontoptions: Ligatures=TeX sansfontoptions: Ligatures=TeX,Scale=MatchLowercase monofontoptions: Scale=MatchLowercase,Scale=0.9

Biblatex

biblatex: true biblio-style: "gost-numeric" biblatexoptions:

- parentracker=true
- backend=biber
- hyperref=auto
- language=auto
- autolang=other*
- citestyle=gost-numeric

Pandoc-crossref LaTeX customization

figureTitle: "Рис." listingTitle: "Листинг" lofTitle: "Список иллюстраций" lolTitle: "Листинги"

Misc options

indent: true header-includes:

- \usepackage{indentfirst}
- \usepackage{float} # keep figures where there are in the text
- \floatplacement{figure}{H} # keep figures where there are in the text

Цель работы

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

Задание

1. Используя команды getopts grep, написать командный файл, который анализирует командную строку с ключами:

- -iinputfile — прочитать данные из указанного файла;
- -ooutputfile — вывести данные в указанный файл;
- -pшаблон — указать шаблон для поиска;
- -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. Командный 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 N (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).

Теоретическое введение

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек:

- оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;
- C-оболочка (или csh) — надстройка на оболочкой Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд;
- оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна;
- BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation). POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна. Рассмотрим основные элементы программирования в оболочке bash. В других оболочках большинство команд будет совпадать с описанными ниже.

Выполнение лабораторной работы

1. Создаю первый исполняемый файл *1.sh* и открываю редактор *emacs*. (рис. [-@fig:001])

создание файла{ #fig:001 width=70% }

3. Используя команды getopts grep, пишу командный файл, который анализирует командную строку с ключами:

- -iinputfile — прочитать данные из указанного файла;
- -ooutputfile — вывести данные в указанный файл;
- -pшаблон — указать шаблон для поиска;
- -C — различать большие и малые буквы;
- -n — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом -p. (рис. [-@fig:002]) (рис. [-@fig:003])

написание скрипта{ #fig:002 width=70% }

написание скрипта{ #fig:003 width=70% }

4. Добавляю право на выполнение файла и проверяю его работу. (рис. [-@fig:004])

проверка первого файла{ #fig:004 width=70% }

5. Создаю второй исполняемый файл *2.sh* и *2.c* и открываю редактор *emacs*. (рис. [-@fig:005])

создание файла{ #fig:005 width=70% }

6. Пишу 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. (рис. [-@fig:006]) Командный файл вызывает эту программу и, проанализировав с помощью команды $?, выдает сообщение о том, какое число было введено. (рис. [-@fig:007])

написание 2.c{ #fig:006 width=70% }

написание 2.sh{ #fig:007 width=70% }

7. Добавляю право на выполнение файла и проверяю его работу. (рис. [-@fig:008])

проверка второго файла{ #fig:008 width=70% }

8. Создаю третий исполняемый файл *3.sh* и открываю редактор *emacs*. (рис. [-@fig:009])

создание файла{ #fig:009 width=70% }

9. Пишу командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N (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). (рис. [-@fig:010])

написание скрипта{ #fig:010 width=70% }

10. Добавляю право на выполнение файла и проверяю его работу. (рис. [-@fig:011])

проверка третьего файла{ #fig:011 width=70% }

11. Создаю четвертый исполняемый файл *4.sh* и открываю редактор *emacs*. (рис. [-@fig:012])

создание файла{ #fig:012 width=70% }

12. Пишу командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицирую его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использую команду find). (рис. [-@fig:013])

написание скрипта{ #fig:013 width=70% }

13. Добавляю право на выполнение файла и проверяю его работу. (рис. [-@fig:014])

проверка четвертого файла{ #fig:014 width=70% }

Выводы

Выполняя данную лабораторную работу я изучил основы программирования в оболочке ОС UNIX/Linux и научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Список литературы{.unnumbered}

::: {#refs} :::