Лабораторная работа №3 по дисциплине “Современные системы программирования”

Антон Кабыш

Задание 1

Цели работы:

- Написать программу работающую текстовую последовательность из файла.
- Получить навыки работы с файловой системой в Java.

Вариант 1

Напишите программу, которая читает несколько строк текста и печатает таблицу, показывающую, сколько раз в тексте встречаются однобуквенные слова, двухбуквенные слова, трехбуквенные слова и т.д.

Вариант 2

Напишите программу, которая вводит строку текста, разбивает ее на лексемы выводит лексемы в обратном порядке.

Вариант 3

Напишите программу выдачи перекрестных ссылок, т.е. программу, которая печатает список всех слов документа и для каждого из этих слов печатает список номеров строк, в которые это слово входит.

Вариант 4

Напишите программу, которая печатает слова из файла, расположенные в порядке убывания частоты их появления. Перед каждым словом напечатайте число его появлений.

Вариант 5

Напишите программу сравнения двух файлов, которая будет печатать первую строку и позицию символа, где они различаются.

Вариант 6

Напишите программу печати на экран набора файлов, которая начинает каждый новый файл с новой страницы и печатает для каждого файла заголовок и счетчик текущих страниц. Имена файлов должны задаваться в командной строке программы.

Вариант 7

В файле большого размера записан некоторый текст. Требуется записать в другой файл его копию в обратном порядке символов. Для обработки текста использовать "окно"в памяти компьютера размером 1024 байт.

Вариант 8

Необходимо подсчитать число цифр в тесктовом файле. Локализовать и вывести на экран строку, содержащий цифру с порядковым номером n/2, где n - общее количество подсчитанных цифр.

Вариант 9

Напишите программу, которая использует генерацию случайных чисел для создания предложений. Программа должна использовать 4 массива строк, называемые `noun` (существительные), `adjective` (прилагательные), `verb` (глаголы) и `preposition` (предлоги).

Программа должна создавать предложение, случайно выбирая слова из каждого массива в следующем порядке: `noun, verb, adjective, preposition noun`. Как только слово выбрано, оно должно быть подсоединено к предыдущему слову в массиве, который достаточно велик для того, чтобы вместить все предложение.

Слова должны быть разделены пробелами. При выводе окончательного предложения, оно должно начинаться с заглавной буквы и заканчиваться точкой. Программа должна генерировать 20 таких предложений.

Задание 2

Цели работы:

- Научится писать качественные консольные утилиты, обрабатывать ввод пользователя и ключи работы программы.
- Получить навыки работы с файловой системой в Java.

Задание:

- Реализовать на Java следующую консольную утилиту.
- Проект упаковать в один Jar с запуском в один клик.
- Предоставить .bat файл запуска.

Вариант 1 — `cat`

На вход утилите `cat` подается список файлов. Утилита считывает их по одному и выводит в стандартный вывод, таким образом, объединяя их в единый поток. Если вместо имени файла указано —, то `cat` читает данные из стандартного ввода до тех пор, пока пользователь не прервет сеанс ввода нажав ввод.

Формат использования:

```
cat [файл1] [файл2]..
```

Пример использования:

```
cat a.txt b.txt
```

Выводит на экран содержимое текстовых файлов:

```
cat a.txt — b.txt > abc.txt
```

Читает содержимое файла a.txt, читает из консоли (-), читает из файла b.txt и записывает вывод в файл abc.txt

Вариант 2 — tail

Утилита tail выводит несколько (по умолчанию 10) последних строк из файла.

Формат использования

```
tail [-n] file
```

Ключ -n <количество строк> (или просто -<количество строк>) позволяет изменить количество выводимых строк. **Пример использования**

```
tail n -20 app.log
tail -20 app.log
```

Выводит 20 последних строк из файла app.log.

Для решения задачи подойдет класс java.io.RandomAccessFile, реализующий произвольный доступ к файлу (чтение и запись с любой позиции в файле).

Вариант 3 — head

Утилита head выводит несколько (по умолчанию 10) первых строк из файла.

Формат использования:

```
head [—n] file
```

Ключ -n <line numbers> (или просто —<line numbers>) позволяет изменить количество выводимых строк.

Пример использования:

```
head n -20 app.log
head -20 app.log
```

Выводит 20 первых строк из файла app.log.

Для решения задачи подойдет класс java.io.RandomAccessFile, реализующий произвольный доступ к файлу (чтение и запись с любой позиции в файле).

Вариант 4 — `nl`

Утилита `arg1nl` выводит переданный файл в стандартный вывод или в другой файл, выполняя нумерацию его строк. Если файл не задан или задан как —, читает стандартный ввод.

Формат использования:

```
nl [-i] [-l] [-n] входной_файл [выходной_файл]
```

- `-i` ЧИСЛО
  Задает шаг увеличения номеров строк
- `-l` ЧИСЛО
  Заданное *число* пустых строк считать одной
- `-n` ФОРМАТ
  Использовать заданный *формат* для номеров строк. формат задается аналогично как в `printf`.

Пример использования

```
nl -i 2 -l 2 -n 4d in.txt
```

Обрабатывает файл `in.txt`, выводит результат в стандартный вывод, инкремент счетчика равен двум (`—i 2`), если встречается подряд две и более пустых строк, то инкремент не выполняется (`—l 2`); при форматировании, номер строки будет занимать четыре символа (`—n 4d`).

Вариант 5 — `cp`

Утилита `cp` копирование файла из одного каталога в другой. Исходный файл остаётся неизменным, имя созданного файла может быть таким же, как у исходного, или измениться.

Формат использования:

```
cp [-f][-i][-n] [директория]исходный_файл [директория]целевой_файл
```

- `-f`
  Разрешает удаление целевого файла, в который производится копирование, если он не может быть открыт для записи.
- `-i` Утилита будет запрашивать, следует ли перезаписывать конечный файл, имя которого совпадает с именем исходного. Для того, чтобы перезаписать файл, следует ввести **y** или его эквивалент. Ввод любого другого символа приведёт к отмене перезаписи данного файла.

- `-n` Не перезаписывать существующий файл (отменяет предыдущий параметр `-i`).

Пример использования:

```
cp -fn src.txt dest.txt
```

Копирует содержимое из `src.txt` в `dest.txt` с ключами `—f` и `—n`.

Вариант 6 — `split`

Утилита `split` копирует и разбивает файл на отдельные файлы заданной длины. В качестве аргументов ей надо указать имя исходного файла и префикс имен выходных файлов. Если файл не задан или задан как —, программа читает стандартный ввод. По умолчанию размер части разбиения равен `1000` строк, а префикс равен `x`.

Имена выходных файлов будут составляться из этого префикса и двух дополнительных букв `aa`, `ab`, `ac` и т. д. (без пробелов и точек между префиксом и буквами). Если префикс имен файлов не задан, то по умолчанию используется `x`, так что выходные файлы будут называться `xaa`, `xab` и т. д.

Формат использования:

```
split [-b | -l] [-d] [входной_файл [префикс_выходных_файлов]]
```

где ключи имеют следующее значение:

- `-b`, `--bytes=num`
  Записывать в каждый выходной файл заданное число `num` байт. При задании числа байт можно использовать суффиксы: `b` означает байты, `k` — `1kb`, `m` — `1Mb`.
- `-l`, `--lines=num`
  Записывать в каждый выходной файл `num` строк.
- `-d`, `--numeric—suffixes`
  Использовать числовые, а не алфавитные суффиксы, начинающиеся с `00`. Суффиксы файлов будут иметь вид: `00`, `01`, `02` и т. д.

Вариант 7 — `uniq`

Утилита `uniq` отфильтровывает повторяющиеся строки во входном файле. Если входной файл задан как — или не задан вовсе, то чтение производится из стандартного ввода. Если выходной файл не задан, запись производится в стандартный вывод.

Если одна и та же строка встречается второй и более разы, то она не записывается в вывод программы.

Формат использования:

```
uniq [-c | -d | -u] [-i] [входной_файл [выходной_файл]]
```

где ключи имеют следующее значение:

- `-u`
  Выводить только те строки, которые не повторяются на входе.
- `-d`
  Выводить только те строки, которые повторяются на входе.
- `-c`
  Перед каждой строкой выводить число повторений этой строки на входе и один пробел.
- `-i`
  Сравнивать строки без учёта регистра.

Вариант 8 — `paste`

Утилита paste выполняет слияние строк/столбцов из файлов и выводит результат в стандартный вывод.

Формат использования:

```
paste [options] [file1 [file2]..]
```

где ключи имеют следующее значение:

- `-s`
  Меняет положение строк со столбцами;
- `-d` разделитель
  Меняет разделитель на указанный (по умолчанию TAB).

Примеры использования
Пусть дан файл **names.txt** со следующим содержимым:

И файл `numbers.txt` с содержимым:

```
555—1234
555—9876
555—6743
867—5309
```

Тогда, применение к ним команды `paste` даст следующий результат:

```
paste names.txt numbers.txt
Mark Smith  555—1234
Bobby Brown 555—9876
Sue Miller  555—6743
Jenny Igotit    867—5309
```

Примение ключа `-s` изменяет вывод программы на горизонтальный:

```
paste -s names.txt numbers.txt
Mark Smith  Bobby Brown Sue Miller  Jenny Igotit
555—1234    555—9876    555—6734    867—5309
```

Использование опции `-d` позволяет задать используемые разделители:

```
paste -d ., names.txt numbers.txt
Mark Smith.555—1234
Bobby Brown,555—9876
Sue Miller.555—6743
Jenny Igotit,867—5309
```

Используя оба ключа:

```
paste —s —d '\t\n' names.txt
Mark Smith      Bobby Brown
Sue Miller      Jenny Igotit
```

Вариант 9 — `join`

Утилита `join` объединяет строки двух упорядоченных текстовых файлов на основе наличия общего поля. По своему функционалу схоже с оператором `JOIN`, используемого в языке SQL для реляционных баз данных, но оперирует с текстовыми файлами.

Команда `join` принимает на входе два текстовых файла и некоторое число аргументов. Если не передаются никакие аргументы командной строки, то данная команда ищет пары строк в двух файлах, обладающие совпадающим первым полем (последовательностью символов, отличных от пробела), и выводит строку, состоящую из первого поля и содержимого обоих строк.

Ключами `-1` или `-2` задаются номера сравниваемых полей для первого и второго файла, соответственно.
Если в качестве одного из файлов указано – (но не обоих сразу!), то в этом случае

вместо файла считывается стандартный ввод.

Формат использования:

```
join [—1 номер_поля] [—2 номер_поля] файл1 файл2 [файл3]
```

Параметры:

- `-1 fiedl_num`
  Задает номер поля в строке для первого файла, по которому будет выполняться соединение.
- `-2 field_num`
  Задает номер поля в строке для второго файла, по которому будет выполняться соединение.

Аргументы:

- **файл1**, **файл2** — входные файлы
- **файл3** — выходной файл, куда записывается результат работы программы.

Примеры использования: Пусть задан файл `1.txt` со следующим содержимым:

```
1 abc
2 lmn
3 pqr
```

И файл `2.txt` со следующим содержимым:

```
1 abc
3 lmn
9 opq
```

Тогда, выполнение команды

```
join 1.txt 2.txt
```

Даст следующий результат:

```
1 abc abc
3 pqr lmn
```

Поскольку в обоих файлах есть строки, чьё первое поле совпадает, это — `1` и `3`.
Выполнение команды

```
join -1 2 -2 2 1.txt 2.txt
```

даст результат

```
abc 1 1
lmn 2 3
```

поскольку теперь сравнение выполняется по 2-му полю для первого и второго файла соответственно.