**Операционные системы и среды  
Л.р.3. Основы программирования на C под Unix. Инструментарий программиста в Unix.**

**Цель:**

Изучение среды программирования и основных инструментов: компилятор/сборщик («коллекция компиляторов») gcc, управление обработкой проекта make (и язык makefile), библиотеки и т.д.   
Практическое использование основных библиотек и системных вызовов: ввод-вывод и работа с файлами, обработка текста, распределение памяти, управление выполнением и т.п.

**Практическая часть**

**Общая постановка задачи:**

Написать программу в соответствии с вариантом задания, создать makefile для управления обработкой проекта и проверить выполнение описанных в нем целей, собрать и протестировать исполняемый файл.

Проект желательно строить многомодульным (например, головной модуль и 1-2 подключаемых к нему модулей с «рабочими» функциями).

Для программ-фильтров надо реализовать возможность явного указания выходного файла в командной строке при вызове, а также опций (если они предусмотрены)

Среди целей *makefile* должны быть сборка и «очистка» (удаление промежуточных файлов) проекта, а также по возможности тестирование исполняемого файла с заранее заготовленными входными данными.

**Варианты заданий:**

1) Инвертирующий фильтр (для потока)  
2) Инвертирующий фильтр (для строк)  
3) Инвертирующий фильтр (для символов)  
4) Фильтр морзянки (прямой)  
5) Фильтр морзянки (обратный)  
6) Криптографический фильтр  
7) Фильтр-«автокорректор»  
8) Фильтр-«автоцензор»  
9) Разворачивание регулярных выражений  
10) Аналог *ps*11) «Записная книжка» («командная»)  
12) «Записная книжка» (интерактивная)

**1 Инвертирующий фильтр (для потока)**

Инверсия порядка байт в потоке (первый → последний, второй → предпоследний, и т.д.).

Надо как-то продумать проблему ограничения объема обрабатываемых данных: поток в общем случае может быть бесконечным, для учебной программы допустимо ввести искусственное ограничение.

**2 Инвертирующий фильтр (для строк)**

Инверсия порядка строк потока (предполагается, что поток – текст, состоящий из отдельных строк). Длину строк можно считать ограниченной некоторой достаточно большой константой.

**3 Инвертирующий фильтр (для символов)**

Инверсия порядка символов в каждой строке потока, порядок самих строк не изменяется. Длину строк можно считать ограниченной некоторой достаточно большой константой.

**4 Фильтр морзянки (прямой)**

Преобразование символов потока в комбинации азбуки Морзе. Непреобразуемые символы отбрасываются.

**5 Фильтр морзянки (обратный)**

Преобразование комбинаций азбуки Морзе в печатные символы. Нераспознанные комбинации снабжаются соответствующими пометками и передаются в выходной поток

Эти два задания могут быть объединены в одной программе, конкретная функция выбирается опцией при вызове. Также эффективно кооперирование исполнителей, хотя бы для тестирования программами друг друга.

**6 Криптографический фильтр**

Шифрование и дешифрование потока. Криптоалгоритм выбирается произвольно (можно простейшие). Способ передачи пароля и других параметров шифрования выбирается произвольно.   
(В случае достаточно сложного криптоалгоритма задание может быть разбито на два, по аналогии с предыдущими: прямое и обратное преобразование.).

Примечание о кооперации и тестировании аналогично предыдущим заданиям

**7 Фильтр-«автокорректор»**

Проверка и коррекция текста по словарю. Проверка производится на совпадение, без анализа словоформ; корректировать достаточно одиночные ошибки. Словарь – внешний файл, его размер заранее не известен.

**8 Фильтр-«автоцензор»**

Проверка и «цензура» текста: удаление (замена фиксированным шаблоном) «запрещенных» слов, заданных словарем. Проверка на простое совпадение, без анализа словоформ. Словарь – внешний файл, его размер заранее не известен.

Опционально – можно предусмотреть управление словарем: добавление, удаление слов, проверка наличия (близко к заданиям «записная книжка»)

**9 Разворачивание регулярных выражений**

Генерация множества строк, заданных регулярным выражением.  
Ввиду сложности «диалект» регулярного выражения сильно ограничивается – например, до поддержки только символьных классов («**[…]**») и «необязательных» символов.   
Входное регулярное выражение поступает в поток ввода, сгенерированные строки – в поток вывода или файл, т.е. получается тоже фильтр.

Возможное практическое использование: подбор частично известного пароля. Если не известны (или известны неточно) лишь несколько (немного) символов пароля, его «ручной» подбор вполне возможен, но процесс перебора вариантов желательно систематизировать.

**10 Аналог *ps***

Упрощённый аналог.   
Запуск программы с аргументом **-l** выводит список процессов (pid и имя исполняемого файла).   
Запуск с аргументом **-p <pid>** выводит дополнительную информацию о заданном процессе.

**11 «Записная книжка» («командная»)**

Работа с «записной книжкой». Программа вызывается на 1 операцию, задаваемую аргументами командной строки, после ее исполнения управление возвращается в shell.

«Записная книжка» (фактически упрощенная база данных) – текстовый файл, записи включают несколько полей (формат выбирается произвольно). Реализуемые операции: добавление, удаление, поиск (по различным полям) и отображение, изменение записи.

**12 «Записная книжка» (интерактивная)**

Аналогично, но режим работы интерактивный (диалоговый): ввод команды, исполнение, ожидание следующей.