Первое задание:
1. Ввести со стандартного ввода произвольное количество строк произвольной длины и расположить их в массиве указателей на char.
2. Отсортировать строки лексикографически (по алфавиту) любым известным методом, меняя местами только указатели
3. Вывести результат столбиком в стандартный вывод.

Требования к программе: программа должна
- содержать минимум **три** функции (не считая main()): ввода динамической строки, сортировки массива строк, вывода результата.
- компилироваться без предупреждений (warnings);
- корректно сортировать входные данные;
- корректно обрабатывать пустые строки;
- не падать (segmentation fault) вне зависимости от объема входных данных;
- быть правильно оформленной (структурные отступы);
- выделенная динамическая память должна освобождаться (free).

Рекомендуемые способы тестирования:

```
$ ./mysort <mysort.c  # проверять глазами

$ ./mysort </etc/passwd >p1
$ sort </etc/passwd >p2
$ diff -u p1 p2  # должны совпадать
$ valgrind   --leak-check=full  -v  ./mysort
```

Для отладки программ рекомендуется использовать отладчик gdb; об основах его использования (а также о других полезных вопросах, которые так или иначе полезно будет знать для выполнения заданий практикума) можно почитать в методичке А. В. Столярова: http://www.stolyarov.info/books/pdf/unixref.pdf

Это не "обязательное к прочтению", но полезное пособие.

Срок: 31 сентября

Второе задание:
Перекодировка файлов из кодировки UTF-8 в UTF-16 и обратно.
Постановку задачи – см. приложение.

Срок: **14 октября**

Третье задание:
Написать программу создания частотного словаря.
Постановку задачи – см. приложение.

Срок: 28 октября

Четвертое задание (ОСНОВНОЕ ДЛЯ ЗАЧЕТА):

"Моделирование работы интерпретатора SHELL" - см. приложение.

Сроки: 5 ноября – 8 декабря

Задание 4 следует сдавать поэтапно:

5 ноября

- реализация альтернативного ввода команды для myshell – с клавиатуры или из файла;
- реализация разбора введенной строки (разбиение на слова и сохранение слов в удобном внутреннем представлении).

Требования к выполнению первого этапа:

1) программа должна выполнять чтение строк (со стандартного ввода или из файла) **в цикле**. В каждой строке необходимо выделить и напечатать столбиком отдельные слова. При этом:

- любое количество идущих подряд пробельных символов обрабатывается так же, как один пробел.
- текст, заключенный в двойные кавычки, рассматривается как одно слово или часть слова, то есть, внутри двойных кавычек пробельные символы рассматриваются как обычные символы. Например:

вход> `aaa "bbb ccc" ddd`
результат:
```
aaa
bbb ccc
ddd
```

вход> `aaaa "bbb"ccc"ddd" eee`
результат:
```
aaaa
bbbcccddd
eee
```

2) допускаются строки произвольной длины, то есть, программа должна вести себя корректно вне зависимости от того, какой длины строка подана на ввод (!).
3) программа завершает работу в ситуации "конец файла" на стандартном вводе (или в файле). Обработка конца файла должна быть реализована корректно.
4) как **отдельные слова** выделяются управляющие символы myshell, которые также выполняют роль разделителей слов (не требуют вокруг себя пробелов):
&, &&, |, ||. ;, >, >>, <, (,) .

Требование к коду: после считывания очередной строки должен быть сформирован **массив** строк(слов)или **список** полученных слов, и только после этого слова должны выводиться на экран, чтобы продемонстрировать корректную работу.

12 ноября

- реализация выполнения очередной команды,
- отслеживание и удаление «зомби»

Требования к выполнению второго этапа:

1)Программа должна уметь выполнять **в цикле** произвольные команды, заданные в командной строке (рассматриваем каждую введенную строку (или строку заданного файла) как команду myshell, воспринимая первое полученное слово как имя команды, остальные - как аргументы команды).

2) Учесть, что процессы, запускаемые на выполнение, находятся на диске в директориях, перечисленных в переменной PATH, или для них указываются полные имена (с путем).

2) Дополнить программу **встроенной** командой cd для смены текущего каталога (то есть, выполнение этой команды надо запрограммировать самостоятельно с помощью вызова функции chdir(const char *), cd - это не процесс, нельзя пользоваться системными вызовами семейства exec()).

Если в команде cd не задан параметр, то переходить надо в домашнюю директорию, имя которой возвращает функция getenv("HOME").

19 ноября (3 этап)

- реализация перенаправления ввода-вывода запускаемых процессов,
- реализация конвейера из n процессов.

26 ноября

- реализация запуска процессов в фоновом режиме

Внимание! Все предыдущие возможности myshell **необходимо реализовать на 3!**

2 декабря

- **на 4 –** реализация анализа статуса завершения запускаемых процессов,
- реализация операций myshell ||, &&, ; .

Замечания:

- *операция ; разделяет две независимые команды, которые выполняются последовательно,*
- *операции || и && имеют одинаковый приоритет и левоассоциативны. Второй процесс в связке с операцией || (&&) запускается только тогда, когда первый завершился с системным или пользовательским кодом завершения, не равным 0 (оба равны нулю). При запуске процессов, связанных логическими операциями, всегда учитывается статус завершения последнего выполнившегося процесса.*

- **на 3 –** сдача предыдущих пунктов.

9 декабря

- **на 5 –** сдача полного myshell со скобками
- **на 4 –** сдача полного myshell без скобок
- **на 3 –** сдача полного myshell без скобок, анализа статуса завершения, реализации операций ||, && , ; .

Шестое задание:

По выбору решить одну из следующих задач на использование IPC средств:

Очереди сообщений.

1. (5 баллов) Написать две программы, одна из которых посылает сообщения другой. Получатель распечатывает содержимое сообщения. Написать программы так, чтобы они не исполнялись родственными процессами.

2. (10 баллов) Написать две программы, отправитель и получатель. Отправитель принимает текст с клавиатуры и рассылает сообщения нескольким копиям программы получателя. Получатель распечатывает свое имя (argv[0]) и содержимое сообщений. Отправитель продолжает работу пока не встретит конец файла. Тогда он рассылает всем получателям сообщение-ограничитель. Как только получатель заканчивает свою очередь, он отправляет сообщение отправителю. После получения такого подтверждения от всех получателей, отправитель удаляет очередь.

3. (10 баллов) Написать две программы, общающиеся через очередью Одна программа – мастер – получает сообщения, генерируемые другой программой - отправителем (текст + любая идентифицирующая информация). Работают n копий отправителя. Мастер распечатывает сообщения (указывая перед ним информацию, идентифицирующую отправителя). Перед завершением работы отправитель посылает сообщение-ограничитель. После получения такого сообщения от всех отправителей мастер удаляет очередь.

Семафоры.

1. (15 баллов) Необходимо смоделировать производственную линию, производящую Продукт. Каждый продукт состоит из детали С и модуля_1. Модуль_1 состоит из детали А и детали В. Изготовление детали А требует 2 с., детали В – 3 с., детали С – 4 с. Элементы производственной линии должны быть представлены процессами. Использовать набор семафоров, по одному для каждой детали и модуля. Как только деталь или модуль произведены, добавляйте единицу к соответствующему семафору. Когда объект используется на следующем этапе, вычитать единицу.

Разделяемая память.

1. (10 баллов) Написать две программы Производитель и Потребитель. Производитель заполняет буфер в разделяемой памяти, а Потребитель читает его. Производитель должен помещать данные в буфер только после того, как Потребитель прочитает его. (для синхронизации можно использовать два семафора – один для записи, другой для чтения).

2. (15 баллов) Реализовать кольцевую очередь фиксированной длины в разделяемой памяти. Программа производитель читает с клавиатуры и помещает прочитанный текст в конец очереди, потребитель берет записи из начала очереди. При обнаружении некоторого условия окончания работы, производитель перед выходом должен поместить его в очередь, когда потребитель получит это сообщение, он завершает работу.

3. (15 баллов) Пусть исполняется n процессов. Часть из них читает, а остальные пишут в разделяемую память. Несколько читающих процессов могут работать с буфером одновременно, но если один из процессов выполняет запись, все остальные ждут. Кроме того, если пишущий процесс хочет обновить запись, он должен дождаться окончания работы процессов-читателей (использовать семафоры для взаимного исключения).

Сроки: 16 декабря

Некоторые обязательные требования ко всем вариантам задания:

- любые занятые ресурсы (динамически выделенная память, средства IPC) должны быть освобождены;

- программы не должны исполняться родственными процессами;

- программы не должны использовать активное ожидание (например, проверка наличия входных данных в цикле), для синхронизации нужно использовать любые (если не указаны в задании) средства IPC.