Модельная реализация программы less с добавленой функциональностью программ cat и sed

Собственно задание

Требуется написать свою реализацию программы less, которая позволяет отображать файлы в терминале, при этом визуазилизировать перемещения по нему. Также данная реализация должна учитывать, что стандартный поток вывода или ввода и вывода одновременно, могут оказаться не в терминале. В этой ситуации программа должна работать как программа cat, то есть печатать текст из файла как есть, без какого либо дополнительного форматирования.

Программа на вход принимает имя файла. Перед именем файла может встретиться параметр -n, который означает, печатать номера строки слева перед текстом (в случае вывода не в терминал данный ключ игнорировать). Так же может встретиться параметр -v, по которому должна быть напечатана версия данной программы и параметр -h, который позволяет распечатать некоторую инструкцию по использованию программы и параметрах, которые ей можно передать. В случае указания -v или -h отображение файла, если он был указан не происходит. Разбирать параметры, переданные программе на вход необходимо с применением функции getopt (man 3 getopt (#include <getopt.h>)).

Перемещение по тексту управляется клавишами стрелочек таким образом, что нажатие вверх/вниз приводит к переходу отображаемой части текста на предыдущую/последующую строчку текста. А влево-вправо — перемещение по позиции в строке. Окно терминала, в котором отображается текст не должно перемещаться вправо дальше последнего символа в самой длинной строке текста (если в тексте встречаются символы код которых занимает больше одной буквы (например русский язык в кодировке UTF8) то нужно учесть актуальный отображаемый размер текста (рекомендуется использовать тип данных wchar_t и функции fgetwc, wcslen, wprintf)). В последней строчке окна терминала должно отображаться «**номер_строки_курсора(всего_строк):**» что является приглашением ко вводу команды.

Если были «включены» номера строк, то при прокрутке текста влево-вправо номера строк должны попрежнему отображаться слева, а непосредственно перед цифрам значок '<', который говорит о том, что слева есть текст. В позиции окна «край слева» вместо знака '<' отображать знак '|'. После цифр должен идти знак ':', но так, чтобы перед ним было вставлено нужное количество пробелов, с учётом что отображение числа как набора символов имеет разный размер. Таким образом далее, после двоеточия идёт столбец текста (сквозной на весь файл). Если номера строк выключены, то „|“ и „>“, остаются, а цифры и двоеточие не отображаются. Если вывод происходит не в терминал, то вообще никакие дополнительные символы не отображаются, в том числе командная строка внизу окна терминала, и все сообщения об ошибках печатаются в стандартный поток ошибок.

Необходимо реализовать следующие команды:

- ***/подстрока_для_поиска*** – производит поиск в тексте вниз до ближайшей находки. Если текст найден, то «курсор» переводится на данную строчку и позицию начиная с которой обнаружена находка. Первая отображаемая строка в окне терминала сверху – это строка, на которой находится курсор. Если подстрока не найдена, то пишется, «не найдено» и после нажатия «enter» всё возвращается к стандартному отображению, с двоеточием в последней строке окна терминала. Если подстрока для поиска пуста, то поиск производится с последней вводившейся подстрокой, до следующего вхождения. Если ничего не вводилось ранее, то выдать ошибку.

- **subst /строка_образец/на_что_заменять/ –** производит поиск с заменой в тексте. При этом в заменяемой строке может встретиться **\n** – означает разбить строку тексте в этом месте при вставке на 2 строки. Последовательность **\n** в строке образца, означает слепить 2 строчки у которых конец совпадает с образцом до **\n**, а начало следующей с образцом после **\n**. Стоит помнить, что **\n** можно указать более одного раза, таким образом нужно будет потенциально просматривать на много строчек вперёд. При этом «***subst /\n//***» сотрёт все пустые строчки в файле. Так же необходимо предусмотреть экранирование символа / и других важных символов, например „\“ и “/“ подряд экранируют /.
- **число** — означает переместить курсор на позицию задаваемую числом. Строки нумеруются с единицы.
- **write "имя_файла" –** сохраняет возможно модифицированный текст командой subst в файл с указанным именем.

Выход из программы должен происходить по закрытию потока ввода, либо по нажатию „q“, но не в том случае, когда набирается одна из команд.

Некоторые замечания по реализации

Рекомендуется строки, прочитанные из файла хранить в программе как двунаправленный список:

```
struct list_node
{
        char *str;
        struct list_node *prev;
        struct list_node *next;
};

typedef struct
{
        size_t num_elements;
        struct list_node *head;
        struct list_node *tail;
} Biderect_list;
```

Для своей работы программа должна будет перевести режим работы терминала в так называемый «non canonical mode». В этом режиме программе передаётся поток байт из терминала как есть, таким образом не нужно нажимать enter, для того, чтобы получить вводимые с клавиатуры данные. Однако в этом случае действия на управляющие сочетания клавиш придётся задавать самому. Например в случае получения ctrl+d необходимо самому организовать закрытие потока ввода. По завершению работы программы, терминал должен быть возвращён в своё исходное состояние. Для работы с терминалом потребуются следующие системные вызовы: ioctl() (на самом деле в некоторых случаях можно воспользоваться библиотечными функциями, которые сами вызывают ioctl см. man tty_ioctl) и fstat(). Пример на манипуляцию с терминалом UNIX доступен здесь: https://github.com/asalnikov/prac_examples/blob/master/terminal_manipulating/term_mode_change.c

Программа сама должна уметь определять размеры окна терминала предоставляемые операционной системой для этого нужно использовать ioctl.