**Get Next Line**

Чтение строки из файлового дескриптора слишком утомительно

**Глава I — Цели**

Этот проект не только позволит вам добавить удобную функцию в вашу коллекцию, но и познакомит вас с очень интересной концепцией в языке C — **статическими переменными**.

**Глава II — Общие инструкции**

* Ваш проект должен быть написан на C.
* Ваш проект должен соответствовать **Norm**. Если у вас есть дополнительные (bonus) файлы/функции, они также должны соответствовать **Norm**, иначе задание получит **0**.
* Ваши функции **не должны** завершаться сбоем (segmentation fault, bus error, double free и т. д.), кроме случаев неопределенного поведения. В противном случае ваш проект будет считаться неработоспособным и получит **0**.
* Все динамически выделенные участки памяти **должны освобождаться** при необходимости. Утечки памяти **не допускаются**.
* Если задание требует Makefile, он должен компилировать исходные файлы с флагами -Wall -Wextra -Werror, использовать cc, а также **не должен** повторно линковать файлы.
* Makefile должен содержать как минимум следующие правила: $(NAME), all, clean, fclean и re.
* Чтобы добавить **бонусные** части, необходимо включить в Makefile правило bonus, которое добавит все запрещенные в основной части заголовочные файлы, библиотеки или функции.
  + Файлы бонусной части должны иметь суффикс \_bonus.{c/h}, если в задании не указано иное.
  + Основная и бонусная части оцениваются отдельно.
* Если проект **разрешает использовать libft**, необходимо скопировать её исходники и Makefile в папку libft.
* **Тестирование**: рекомендуется писать тестовые программы, хотя они не подлежат сдаче и оценке. Это поможет проверять свою работу и работу других студентов.
* Проект должен быть загружен в **git-репозиторий**, и **только код в репозитории будет оцениваться**.
* Если проверку кода выполняет **Deepthought**, она начнётся **после** оценки со стороны ваших коллег. Если в процессе возникнет ошибка, оценка остановится.

**Глава III — Обязательная часть**

**Название функции:**

c

КопироватьРедактировать

char \*get\_next\_line(int fd);

**Файлы для сдачи:**

* get\_next\_line.c
* get\_next\_line\_utils.c
* get\_next\_line.h

**Параметры:**

* fd — файловый дескриптор для чтения

**Возвращаемое значение:**

* Прочитанная строка → **корректное поведение**
* NULL → если больше нечего читать или произошла ошибка

**Разрешённые функции:**

* read
* malloc
* free

**Описание**

* Функция get\_next\_line() должна **считывать и возвращать по одной строке** из файла, переданного через fd.
* Если больше нечего читать или возникла ошибка — функция должна вернуть NULL.
* Функция должна работать **как с файлами, так и со стандартным вводом (stdin)**.
* Возвращаемая строка **должна включать символ \n**, если он есть в файле. Исключение — если файл закончился без \n.
* Заголовочный файл get\_next\_line.h должен **содержать хотя бы прототип** функции get\_next\_line().
* Вспомогательные функции должны быть вынесены в get\_next\_line\_utils.c.
* **Статические переменные** помогут в реализации.

**Дополнительные требования**

* В get\_next\_line() **должен использоваться буфер** (BUFFER\_SIZE), определяемый при компиляции с помощью флага:

sh

КопироватьРедактировать

-D BUFFER\_SIZE=n

* + BUFFER\_SIZE **будет изменяться** проверяющими для тестирования кода.
* Проект **должен компилироваться** как с флагом -D BUFFER\_SIZE, так и без него.
* Пример компиляции (с BUFFER\_SIZE = 42):

sh

КопироватьРедактировать

cc -Wall -Wextra -Werror -D BUFFER\_SIZE=42 <files>.c

* **Не загружайте весь файл сразу!**
  + Считывайте **только необходимый минимум**.
  + Если встретился символ новой строки (\n), **нужно вернуть текущую строку**.
  + **Не читайте весь файл перед разбором строк**!

**Запрещено**

* **Использовать свою libft**.
* **Использовать lseek()**.
* **Глобальные переменные**.

**Глава IV — Бонусная часть**

Проект достаточно простой, поэтому сложные бонусы не предусмотрены. Однако, если вы закончили обязательную часть, попробуйте реализовать бонусные функции.

**Требования к бонусной части:**

1. **Функция должна использовать только одну статическую переменную**.
2. **Функция должна поддерживать несколько файловых дескрипторов одновременно**.
   * Например, если читается сразу fd = 3, 4, 5, вызов get\_next\_line() для каждого из них должен **не терять состояние** и **не смешивать строки** между разными fd.
3. Файлы бонусной части должны иметь суффикс \_bonus:
   * get\_next\_line\_bonus.c
   * get\_next\_line\_bonus.h
   * get\_next\_line\_utils\_bonus.c

**Внимание!** Бонусная часть будет оцениваться **только если основная часть выполнена идеально**. Если обязательная часть не пройдена, бонусы **не засчитываются**.

**Глава V — Сдача и оценка**

* Проект сдаётся в **git-репозиторий**, и только файлы в нём будут оцениваться.
* **Перед сдачей проверьте** правильность названий файлов!
* **Проверяйте свои тесты**:
  1. **Размер буфера и длина строк могут сильно отличаться**.
  2. **Файловый дескриптор может указывать не только на файлы**.
* **Обменивайтесь тестами с коллегами** и готовьтесь к защите.
* После успешной сдачи не забудьте **добавить get\_next\_line() в свою libft**.