Printf

Резюме: этот проект довольно прост. Вы перекодируете printf. Надеюсь, вы сможете повторно использовать его в будущих проектах, не опасаясь, что вас сочтут читером. В основном вы узнаете, как использовать вариативные аргументы.

Глава I

Вступление

Универсальность функции printf в C представляет для нас отличное упражнение в программировании. Этот проект средней сложности. Это позволит вам открыть для себя вариативные функции в C.

Ключ к успеху ft\_printf - это хорошо структурированный и хорошо расширяемый код.

Глава II.

Общие инструкции

• Ваш проект должен быть написан в соответствии с Нормой. Если у вас есть бонусные файлы / функции, они включены в проверку нормы, и вы получите 0, если внутри есть ошибка нормы.

• Ваши функции не должны завершаться неожиданно (ошибка сегментации, ошибка шины, двойное освобождение и т. Д.), За исключением неопределенного поведения. Если это произойдет, ваш проект будет считаться нефункциональным и получит 0 во время оценки.

• Все пространство памяти, выделенное кучей, должно быть должным образом освобождено при необходимости. Утечки недопустимы.

• Если субъект требует этого, вы должны отправить Makefile, который скомпилирует ваши исходные файлы в требуемый вывод с флагами -Wall, -Wextra и -Werror, и ваш Makefile не должен повторно связываться.

• Ваш Makefile должен содержать как минимум правила $ (NAME), all, clean, fclean и re.

• Чтобы превратить бонусы в свой проект, вы должны включить бонус правила в свой Makefile, который добавит все различные заголовки, библиотеки или функции, запрещенные в основной части проекта. Бонусы должны быть в другом файле \_bonus. {C / h}.

Оценка обязательной и бонусной части проводится отдельно.

• Если ваш проект позволяет вам использовать ваш libft, вы должны скопировать его исходный код и связанный с ним Makefile в папку libft с соответствующим Makefile. Makefile вашего проекта должен скомпилировать библиотеку, используя свой Makefile, а затем скомпилировать проект.

• Мы рекомендуем вам создавать программы тестирования для вашего проекта, даже если эту работу не нужно будет отправлять и оценивать. Это даст вам возможность легко проверить свою работу и работу коллег. Вы найдете эти тесты особенно полезными во время защиты. Действительно, во время защиты вы можете использовать свои тесты и / или тесты партнера, которого вы оцениваете.

• Отправьте свою работу в назначенный репозиторий git. Оцениваться будет только работа в репозитории git. Если DeepThreadt назначен для оценки вашей работы, это будет сделано после ваших оценок коллег. Если в каком-либо разделе вашей работы во время

При выставлении оценок Deep Thinking оценка остановится.

Глава III.

Обязательная часть

Название программы

libftprintf.a

Сдать файлы

\* .c, \* / \*. c, \* .h, \* / \*. h, Makefile

Makefile

all, clean, fclean, re, бонус

Внешние функции.

malloc, free, write, va\_start, va\_arg, va\_copy, va\_end

Libft авторизован

да

Описание

Напишите библиотеку, содержащую ft\_printf, функцию, имитирующую реальный printf

• Прототип ft\_printf должен быть int ft\_printf (const char \*, ...);

• Вам необходимо перекодировать функцию printf библиотеки libc

• Он не должен управлять буфером, как настоящий printf.

• Он будет управлять следующими преобразованиями: cspdiuxX%

• Он будет сравнен с реальным printf

• Вы должны использовать команду ar для создания вашей библиотеки, использование команды libtool запрещено.

Небольшое описание необходимого преобразования:

•% c печатает один символ.

•% s вывести строку символов.

•% p Аргумент указателя void \* печатается в шестнадцатеричном формате.

•% d вывести десятичное число (с основанием 10).

•% i вывести целое число по основанию 10.

•% u вывести десятичное число без знака (с основанием 10).

•% x вывести число в шестнадцатеричном формате (с основанием 16) со строчными буквами.

•% X выводит число в шестнадцатеричном формате (с основанием 16) в верхнем регистре.

• %% вывести знак процента.

для более полных ссылок: man 3 printf / man 3 stdarg

Глава IV.

Бонусная часть

• Если обязательная часть не идеальна, даже не думайте о бонусах.

• Не обязательно использовать все бонусы.

• Управляйте любой комбинацией следующих флагов: ’-0.’ И минимальная ширина поля со всеми преобразованиями.

• Управляйте всеми следующими флагами: ’# +’ (да, один из них - пробел)

Если вы планируете делать бонусы, вам следует подумать о том, как их делать с самого начала, чтобы избежать наивного подхода.