Get Next Line

fdで行を読むのは面倒くさい

概要：このプロジェクトの目的は、ファイル記述子から読み取った行を返す関数をコーディングさせること　です

コンテンツ

第1章 目標

このプロジェクトでは、非常に便利な関数をコレクションに加えることができるだけでなく、C言語のプログラミングにおいて非常に興味深い新しい概念である「静的変数」を学ぶことができます。

第二章

共通事項

- あなたのプロジェクトは規範に沿って書かれていなければなりません。ボーナスファイル/関数がある場合は、それらも規範チェックに含まれ、内部に規範エラーがある場合は0が表示されます。
- 未定義の動作とは別に、関数が予期せず終了する（セグメンテーション・フォールト、バス・エラー、ダブル・フリーなど）ことがあってはなりません。このような現象が発生した場合、プロジェクトは非機能的であるとみなされ、評価の際に0点が付けられます。
- ヒープに割り当てられたメモリ空間は、必要に応じて適切に解放されなければなりません。リークは許されません。
- 課題で要求された場合は、-Wall、-Wextra、-Werrorのフラグでソースファイルを要求された出力にコンパイルするMakefileを提出する必要があり、Makefileは再リンクしてはいけません。
- Makefileには、少なくとも、$(NAME)、all、clean、fcleanのルールが含まれていなければなりません。
  を参照してください。
- プロジェクトにボーナスを回すには、Makefileにルールボーナスを含めなければなりません。このルールボーナスは、プロジェクトのメイン部分で禁止されているすべての様々なヘッダー、librairies、または関数を追加します。ボーナスは別のファイル _bonus.{c/h} に記述する必要があります。必須部分とボーナス部分の評価は別々に行われます。
- あなたのプロジェクトであなたの**libft**を使うことができる場合、そのソースと関連するMakefileを、関連するMakefileのある**libft**フォルダにコピーする必要があります。プロジェクトのMakefileは、そのMakefileを使ってライブラリをコンパイルし、その後、プロジェクトをコンパイルする必要があります。
- テストプログラムの**提出や評価はありませんが、**プロジェクトのテストプログラムを作成することをお勧めします。テストプログラムを作成することで、自分の作品や仲間の作品を簡単にテストすることができます。これらのテストは、特にディフェンス時に役立つでしょう。実際、弁論大会では、自分のテストや評価している仲間のテストを自由に使用することができます。
- 割り当てられた git リポジトリに作品を提出してください。git reposi-toryにある作品のみが採点されます。Deepthoughtがあなたの作品の採点を担当する場合は、あなたの相互評価の後に行われます。Deepthoughtによる採点中に作品のいずれかの

セクションでエラーが発生した場合、評価は中止されます。

第3章

必須項目 - Get_next_line

機能名	get_next_line
プロトタイプ	char *get_next_line(int fd)です。
ファイルの提出	get_next_line.c, get_next_line_utils.c, get_next_line.h
パラメータ	読み込み先のファイルデスクリプタ
戻り値	Read line:　正しい動作 NULL：他に読むべきものがないか、エラーが発生した場合
エクスターナル・ファンクティス	読み取り、malloc、フリー
説明	から読み取った行を返す関数を書いてください。 ファイルディスクリプター

- 関数 get_next_line をループで呼び出すと、ファイル記述子で利用可能なテキストを1行ずつ、最後まで読むことができます。
- この関数は、今読み込んだ行を返す必要があります。他に読むべきものがない場合や、エラーが発生した場合は、NULLを返します。
- ファイルから読み込んだときと、標準入力から読み込んだときに、関数がうまく動作するようにしてください。
- このプロジェクトでは、libftは使用できません。get_next_lineが動作するために必要な関数を含むget_next_line_utils.cファイルを追加する必要があります。
- プログラムのコンパイル時には、-D BUFFER_SIZE=xxというフラグを立ててください。この値は、get_next_lineのreadコールのバッファサイズとして使用されます。この値は、あなたの評価者やmoulinetteによって変更されます。
- このようにして、プログラムがコンパイルされます。
  gcc -Wall -Wextra -Werror -D BUFFER_SIZE=4 <files2>.c.
- ファイルまたは標準入力から読み取るためには、コンパイル時に定義されたBUFFER_SIZEを使用する必要があります。この値は、テストのために評価中に変更されます。

- ヘッダーファイルget_next_line.hには、少なくとも関数get_next_lineのプロトタイプがなければなりません。

Get Next Line （次の行　　を取得する）fdで行を読むのはとても面倒です。

BUFFER_SIZEの値が9999の時も、BUFFER_SIZEの値が1の時も、10000000時も、この関数は動作しますか？

get_next_lineが呼ばれるたびに、できるだけ読み込まないようにしなければなりません。もし、改行があった場合は、現在の行を戻さなければなりません。ファイル全体を読んでから各行を処理するのはやめましょう。

テストをしないでプロジェクトを提出してはいけません。たくさんのテストを実行して、ベースをカバーしましょう。ファイルから、リダイレクトから、標準入力から読んでみてください標準出力に改行を送ると、プログラムはどのように動作しますか？CTRL-Dも？

- lseekは許可された関数ではありません。ファイルの読み込みは一度だけ行わなければなりません。
- get_next_lineは、2回の呼び出しの間に、同じファイル記述子が最初のfdからすべてを読み込む前に別のファイルに切り替わる場合、未定義の動作をすると考えています。
- 最後に、get_next_lineがバイナリファイルからの読み込み時に未定義の動作をすることを考えます。しかし、望むならば、この動作を首尾一貫したものにすることができます。
- グローバル変数の使用は禁止されています。
- 重要：読まれたラインの後には、必ず「\n」をつけて返す必要があります。
ただし、ファイルの最後まで到達していて、「„^ ω^„」がない場合に限ります。

まずは、静的変数とは何かを知ることから始めましょう。
https://en.wikipedia.org/wiki/Static_variable

第4章 ボーナス

パート

get_next_lineというプロジェクトは単純で、ボーナスの余地はほとんどありませんが、あなたには想像力があることでしょう。必須パートをクリアした方は、ぜひこのボーナスパートをクリアして先に進んでください。ただし、必須パートが完璧でない場合は、ボーナスは考慮されませんのでご注意ください。
このパートでは、_bonus.[c\]で終わる3つの必須ファイルをすべて提出してください。

- get_next_lineを単一のスタティック変数で継承すること。
- get_next_lineで複数のファイル記述子を管理できるようにする。例えば、ファイルディスクリプターと3,4読み取りに5アクセスできる場合、それぞれのディスクリプターの読み取りスレッドを失うことなく、3で1回、4で1回、3で1回、5で1回など、get_next_lineを呼び出すことができます。