ミニシェル

貝殻のような美しさ

概要
このプロジェクトは、簡単なシェルを作るというものです。
そう、あなた自身の小さなバッシングです。
プロセスやファイル記述子について多くのことを学ぶことができます。

バージョン：6

内容

第一章 はじめに

シェルの存在は、ITの存在そのものとリンクしているのです。

当時、1/0を揃えてコンピュータと通信するのは、開発者の誰もが*「イライラする」*と口を揃えて言っていた。

そこで、人間の言葉に近い言葉で、コンピュータと対話的にやりとりするソフトウエアを作ろうということになったのだ。

Minishellのおかげで、タイムスリップして、*Windows*が存在しなかった時代に人々が直面していた問題に戻ることができます。

第二章

共通事項

- プロジェクトはC言語で書かれている必要があります。

- あなたのプロジェクトはNormに従って書かれている必要があります。ボーナスファイル/関数がある場合、それらはノームチェックに含まれ、内部にノームエラーがある場合は0が返されます。

- 未定義の動作とは別に、関数が予期せず終了しないこと（セグメンテーションエラー、バスエラー、ダブルフリーなど）。このような場合、プロジェクトは非機能とみなされ、評価時に0点が与えられます。

- ヒープで確保されたメモリ空間は、必要なときに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。

- 対象がそれを要求している場合、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 -Wall、-Wextra、-Werror フラグで提出し、cc を使用し、その 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、および
再

- 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る必要があります。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様々なヘッダー、 librairies または関数をすべて追加します。ボーナスは、主体が何も指定しない場合は、別のファイル _bonus.{c/h} に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。

- プロジェクトで libft を使用できるようにする場合、そのソースと関連する Makefile を libft フォルダにコピーしておく必要があります。あなたのプロジェクトの 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しなければなりません。

- **この作品は提出する必要はなく、採点もされませんが、**プロジェクトのためにテストプログラムを作成することをお勧めします。テストプログラムは、自分の作品や仲間の作品を簡単にテストする機会を与えてくれます。このようなテストは、特にデフェンスの際に役立つと思います。実際、審査では、自分のテストや審査する仲間のテストを自由に使用することができます。

- 指定された git リポジトリに作品を提出する。git リポジトリにある作品だけが採点対象となります。Deepthought があなたの作品の採点を担当する場合、採点は以下のように行われます。

ピアエバリュエーション後Deepthoughtの採点中に、あなたの作品のいずれかのセクションでエラーが発生した場合、評価は停止されます。

第三章 必須項目

プログラム名	ミニシェル
ファイルを提出する	Makefile, *.h, *.c
メイクファイル	NAME、All、 Clean、Fclean、Re
論証	
外部機能。	readline, rl_clear_history, rl_on_new_line, rl_replace_line, rl_redisplay, add_history, printf, malloc, free, write, access, open, read, close, fork, wait, waitpid, wait3, wait4, signal, sigaction, sigemptyset, sigaddset, kill, exit, getcwd, chdir.All rights reserved,stat, lstat, fstat, unlink, execve, dup, dup2, pipe, opendir, readdir, closedir, strerror, perror, isatty, ttyname, ttyslot, ioctl, getenv, tcsetattr, tcgetattr, tgetent, tgetflag, tgetnum, tgetstr, tgoto, tputs
リブート認可	はい
商品説明	シェルを書く

シェルが必要です。

- 新しいコマンドを待っているときにプロンプトを表示する。
- 作業履歴があること。

- 正しい実行ファイルを検索して起動する（PATH変数に基づくか、相対パスまたは絶対パスを使用する）。
- 複数のグローバル変数を使用しない。考えてみてください。その目的を説明する必要があります。
- 閉じていない引用符や、" \"（バックスラッシュ）、" ; "（セミコロン）など対象が必要としない特殊文字は解釈しない。
- シェルが引用列のメタ文字を解釈しないようにする ' (シングルクォート) を処理します。
- シェルが $ (ドル記号) 以外のメタ文字を解釈しないようにする " (二重引用符) を処理します。

- **リダイレクトを**実装する。
  - <は入力をリダイレクトする必要があります。
  - >は出力をリダイレクトする必要があります。
  - << はデリミタを与えて、そのデリミタを含む行が見えるまで入力を読む必要があります。しかし、履歴を更新する必要はない!
  - >> は append モードで出力をリダイレクトする必要があります。
- **パイプ**(|文字)を実装します。パイプラインの各コマンドの出力は、パイプで次のコマンドの入力に接続されます。
- その値に展開されるべき**環境変数**（$の後に一連の文字が続く）を処理します。
- 最も最近実行されたフォアグラウンド・パイプラインの終了ステータスに展開されるべき$?を処理します。
- ctrl-C、ctrl-D、ctrl-Cのような、bashのような動作をするものを扱います。
- インタラクティブモードでは
  - ctrl-Cは新しいプロンプトを新しい行に表示します。
  - ctrl-D でシェルを終了します。
  - ctrl-ja は何もしません。
- お使いのシェルが以下の**ビルトインを**実装している必要があります。
  - オプション -n の付いたエコー
  - 相対パスまたは絶対パスだけを指定したcd
  - オプション無しのpwd
  - 無手順輸出
  - むこうじょうめん
  - 無手順エンベロープ
  - 無手順で終了

readline()関数は、メモリリークを起こすことがあります。それらを修正する必要はありません。しかし、**だからといって、あなた自身のコード、そう、あなたが書いたコードがメモリリークを起こす可能性があるわけでは**ありません。

主題の説明に限定したほうがいい。　　聞かれていないことは、必要ありません。
要件に疑問がある場合は、bashを参考にするとよいでしょう。

第四章 ボーナス

パート

あなたのプログラムは、実装しなければならないのです。

- 優先順位を表す括弧付きの && と || です。
- ワイルドカードの*は、現在の作業ディレクトリに対して機能するはずです。

ボーナスパーツは、必須パーツがPERFECTである場合にのみ査定されます。パーフェクトとは、必須パートが統合的に行われ、誤動作することと なく動作することを意味します。必須条件をすべてクリアしていない場合、ボーナスパーツの評価は一切行われません。

第五章

提出と相互評価

通常通り、Gitリポジトリに課題を提出する。ディフェンスでは、あなたのリポジトリ内の作品だけが評価されます。ファイル名が正しいかどうか、遠慮なく再確認してください。