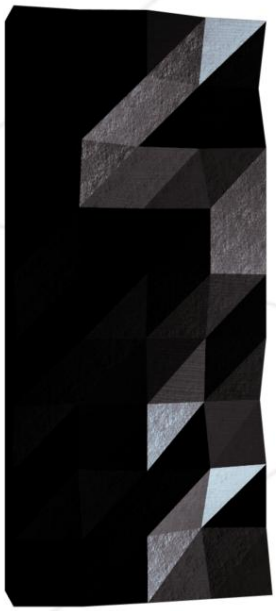

Минишелл

Красивый, как ракушка

Описание:
Этот проект посвящен созданию простой оболочки.
Да, твоя маленькая тусовка.
Вы узнаете много нового о процессах и файловых дескрипторах.

Версия: 6

Содержание

Глава I

Вступление

Существование оболочек связано с самим существованием ИТ.

В то время все разработчики соглашались с тем, что общение с компьютером с помощью выровненных Переключатели 1/0 серьезно раздражали.

Вполне логично, что им пришла в голову идея создать программу для общения с компьютером с помощью интерактивных строк команд на языке, несколько близком к человеческому.

Благодаря Minishell вы сможете путешествовать во времени и возвращаться к проблемам люди сталкивались, когда Windows не существовало.

Глава II

Общие инструкции

- Ваш проект должен быть написан на C.
- 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 неработоспособным и получит 0 баллов при оценке.
- При необходимости все пространство памяти, выделенное в куче, должно быть должным образом освобождено. Нет утечек будут терпеть.
- Если субъект требует этого, вы должны отправить Makefile, который скомпилирует ваши исходные файлы в требуемый результат с флагами -Wall, -Wextra и -Werror, используйте cc, и ваш Makefile не должен перекомпоновываться.
- Ваш Makefile должен как минимум содержать правила $(NAME), all, clean, fclean и re.
- Чтобы включить бонусы в свой проект, вы должны включить бонусное правило в свой Makefile, которое добавит все различные заголовки, библиотеки или функции, запрещенные в основной части проекта. Бонусы должны быть в другом файле _bonus.{c/h}, если в теме не указано ничего другого. Обязательная и бонусная части оцениваются отдельно.
- Если ваш проект позволяет вам использовать ваш libft, вы должны скопировать его исходники и связанный с ним файл Makefile в папку libft с соответствующим файлом Makefile. Makefile вашего проекта должен скомпилировать библиотеку с помощью Makefile, а затем скомпилировать проект.
- Мы рекомендуем вам создать тестовые программы для вашего проекта, даже несмотря на то, что эту работу не нужно будет отправлять и она не будет оцениваться. Это даст вам возможность легко проверить свою работу и работу ваших коллег. Вы найдете эти тесты особенно полезными во время вашей защиты. Действительно, во время защиты вы можете использовать свои тесты и/или тесты коллеги, которого вы оцениваете.
- Отправьте свою работу в назначенный репозиторий git. Оцениваться будет только работа в репозитории git. Если Deepthought будет назначен для оценки вашей работы, это будет сделано

после ваших экспертных оценок. Если во время оценивания Deepthought произойдет ошибка в каком-либо разделе вашей работы, оценка будет остановлена.

Глава III

Обязательная часть

Название программы	минишелл
Сдать файлы	Makefile, *.h, *.c
Аргументы	ИМЯ, все, чисто, fчисто, ре
Makefile	
Внешние функции.	строка чтения, rl_clear_history, rl_on_new_line, rl_replace_line, rl_redisplay, add_history, printf, malloc, бесплатно, запись, доступ, открыть, прочитать, близко, вилка, подождите, ожидание, ожидание3, ожидание4, сигнал, sigaction, sigemptyset, sigaddset, kill, exit, getcwd, chdir, stat, lstat, fstat, unlink, execve, дуп, дуп2, труба, opendir, readdir, closeir, strerror, perror, isatty, ttyname, ttyslot, ioctl, getenv, tcsetattr, tcgetattr, tgetent, tgetflag, tgetnum, tgetstr, tgoto, tputs
Либфт авторизован	Да
Описание	Написать оболочку

Ваша оболочка должна:

• Отображение подсказки при ожидании новой команды.

• Иметь трудовой стаж.

• Найдите и запустите нужный исполняемый файл (на основе переменной PATH или с помощью относительный или абсолютный путь).

• Не используйте более одной глобальной переменной. Подумай об этом. Вам придется объяснить его цель.

• Не интерпретировать незакрытые кавычки или специальные символы, которые не требуются тема, такая как \ (обратная косая черта) или ; (точка с запятой).

• Дескриптор ' (одинарная кавычка), который должен предотвратить интерпретацию метаданных оболочкой. символов в цитируемой последовательности.

• Дескриптор " (двойная кавычка), который должен препятствовать интерпретации оболочкой метасимволов в заключенной в кавычки последовательности, за исключением $ (знак доллара).

• Реализовать перенаправления:

< должен перенаправлять ввод.

> должен перенаправлять вывод.

<< следует указать разделитель, затем считывать ввод, пока не будет видна строка, содержащая разделитель. Тем не менее, он не должен обновлять историю!

>> следует перенаправлять вывод в режиме добавления.

• Реализовать трубы (символ |). Вывод каждой команды в конвейере подключен к вводу следующей команды через конвейер.

• Обрабатывать переменные окружения ($ с последующей последовательностью символов), которые должны расширяться до их значений.

• Обрабатывать $? который должен расширяться до состояния выхода самого последнего запущенного конвейера переднего плана.

• Обрабатывайте ctrl-C, ctrl-D и ctrl-\, которые должны вести себя как в bash.

• В интерактивном режиме:

ctrl-C отображает новое приглашение в новой строке.

ctrl-D закрывает оболочку.

ctrl-\ ничего не делает.

• В вашей оболочке должны быть реализованы следующие встроенные модули:

эхо с опцией -n

cd только с относительным или абсолютным путем

pwd без параметров

экспортировать без параметров

снято без параметров

env без опций и аргументов

выйти без вариантов

Функция readline() может вызвать утечку памяти. Вам не нужно их исправлять. Но это не означает, что ваш собственный код, да код, который вы написали, может иметь утечки памяти.

Следует ограничиться описанием предмета. Все, что не задано, не требуется.

Если у вас есть какие-либо сомнения относительно требования, примите bash в качестве ссылки.

Глава IV

Бонусная часть

Ваша программа должна реализовать:

- && и || со скобками для приоритетов.
- Подстановочные знаки * должны работать для текущего рабочего каталога.

Бонусная часть будет оцениваться только в том случае, если обязательная часть будет ИДЕАЛЬНОЙ. Идеальный означает, что обязательная часть была полностью выполнена и работает без сбоев. Если вы не выполнили ВСЕ обязательные требования, ваша бонусная часть вообще не будет оцениваться.

Глава V

Представление и экспертная оценка

Сдайте задание в своем репозитории Git, как обычно. Во время защиты будет оцениваться только работа внутри вашего репозитория. Не стесняйтесь перепроверять имена ваших файлов, чтобы убедиться, что они верны.