ЗАДАНИЕ ПРАКТИКУМА "Модельный Shell-интерпретатор"

214 группа, 3 семестр, 2020 год

Предлагается реализовать под управлением ОС Unix интерактивный командный интерпретатор, осуществляющий в цикле считывание командной строки со стандартного ввода, анализ и исполнение соответствующих действий.

Задание выполняется в несколько этапов.

Первый этап (03-17 ноября)

Написать первый вариант командного интерпретатора, который в цикле выполняет одиночную команду ОС. Программа читает со стандартного ввода строку и разбивает ее на отдельные слова, разделенные пробелами или табуляциями. Любое количество идущих подряд пробельных символов обрабатывается так же, как один пробел. Текст, заключенный в двойные кавычки, рассматривается как одно слово или часть слова, т.е. внутри двойных кавычек пробельные символы рассматриваются как обычные символы. Например:

```
> vi "text with space"

> gcc -o "ccc ddd" t1.c
```

Допускаются строки произвольной длины, т.е. программа должна вести себя корректно вне зависимости от того, какой длины строка подана на ввод (!).

После разбиения строки на слова первое полученное слово воспринимается как команда (то есть имя файла с исполняемой программой), остальные — как параметры команды – то есть аргументы командной строки. Интерпретатор должен запускать указанную команду с указанными аргументами, считая, что файл с исполняемым кодом должен находиться в диске в директориях, перечисленных в переменной PATH, или указан по полному имени — абсолютному или относительному пути.

Две команды - cd для смены текущего каталога, и exit – для выхода из программы, должны быть реализованы как **встроенные** команды, то есть программа должна выполнять из сама, а не вызывать (несуществующие) внешние команды.

Программа завершает работу в ситуации "конец файла" на стандартном вводе или при вводе пользователем команды exit. Обработка конца файла должна быть реализована корректно.

Второй этап (17 ноября – 1 декабря)

Модифицировать чтение команды и аргументов таким образом, чтобы символы '<', '>' или '>>' воспринимались как разделительные. Реализовать перенаправление ввода и вывода, определяемое символами ‘<', '>' или >>'. Модифицировать чтение команды и аргументов таким образом, чтобы символ '|' воспринимался как разделительный. Реализовать запуск команд конвейером. В минимальном варианте достаточно реализовать конвейер из двух команд, в этом случае при наличии в командной строке более чем одного символа '|' нужно выдавать сообщение об ошибке. В полном варианте ограничений на длину конвейера быть не должно.

Третий этап (1 декабря – 8 декабря)

Модифицировать чтение команды и аргументов таким образом, чтобы она воспринимала символ '&' (если он встречен **вне** кавычек) как разделительный символ (т.е. символ, который является отдельным словом сам по себе). Реализовать выполнение команд в фоновом режиме. Запускать в фоновом режиме команды, последним словом в котором является символ '&'. Если символ '&' встречен не в конце, выдавать сообщение об ошибке. По завершении команды, выполнявшейся в фоновом режиме, выдавать сообщение о ее завершении и код завершения.

Четвертый этап (8 декабря – 20 декабря)

Реализовать:

- связку ';' (сначала выполняется одна команда, потом вторая)
- связку '||' (сначала выполняется первая команда, и если она завершилась неудачей, то выполнятся вторая команда)
- связку '&&' (сначала выполняется первая команда, и если она завершилась успешно, то выполняется вторая команда)

Для сдачи на «отлично» требуется добавить следующую функциональность:

- круглые скобки (содержимое скобок, имеющее произвольную сложность, выполняется как отдельно взятая команда). Эту возможность можно выбирать только вместе с одной из трех предыдущих, в противном случае невозможно проверить ее наличие.
- обратные апострофы (подстановка результата выполнения команды).
- просмотр и модификацию переменной PATH