Отчет по программе shell (task4).
(Автор Толстошеев Иван)

Программа shell функционально состоит из двух частей – преобразователя (парсера) и исполнителя. Кроме этого shell включает в себя несколько других впомогательных функций.

Все вводимые команды преобразуются с помощью парсера во внутреннее представление в соотвествии со следующими формулами:

БНФ программы shell

```
<команда SHELL>::=<список команд>
<список команд>::=<конвейер>[{; & || &&} <конвейер>][{; &}]
<конвейер>::=<команда>[|<команда>]
<команда>::=<простая команда>|(<список команд>)
<простая команда>::=
<имя файла> [<аргумент>] [< <имя файла>] [{>> >} <имя файла>]
```

Выражение в квадратных скобках может повторяться один раз или более. Из лексем, находящихся в фигурных скобках выбирается только одна.

Парсер реализован методом рекурсивного спуска и выполнен в виде нескольких функций:

```
struct cmds* parse(char *command)
struct cmds* get_list_of_commands()
struct cmds* get_conveyor()
struct cmds* get_command()
struct cmds* get_simple_command()
```

Эти функции и осуществляют рекурсивный спуск в соотвествии с приведенной выше БНФ. Функция `parse()` получает в аргументе `command` указатель на строку, введенную пользователем и возвращает указатель на начало динамической структуры, представляющую команду.

Динамичекая структура является деревом. Вот описание узла этого дерева:

```
struct cmds { // Структура одной команды
 char** arrargs; // список из имени команды и аргументов
 char *infile; // переназначенный файл стандартного ввода
 char *outfile; //переназначенный файл стандартного вывода
 int appendfile; //флаг, означающий, что вместо >  обнаружено >>.
 int backgrnd; //=1, если команда подлежит выполнению в фоне
 int and;
 int or;
 struct cmds* subcmd; //команды для запуска в дочернем shell
 struct cmds* pipe; //следующая команда после "|"
 struct cmds* next; //следующая после ";" (или после "&")
};
```

`arrargs` – это сылка на динамический массив аргументов команды.

Исполнитель реалиован в виде функции `execute()`. Эта функция рекурсивно обходит все узлы дерева, полученного от преобразователя и и выполняет команды в соотвествии с указанными условиями. После того, как функция `execute()` завершает свою работу управление передается функции `freemem()`. Она освобождает память от ставшего уже ненужного дерева.