Задание практикума: командная оболочка операционной системы

Сальников А.Н.

17 апреля 2015 г.

Содержание

1 Общее описание

Реализовать программу на языке C, осуществляющую частичную эмуляцию командной оболочки (shell). Примеры программных оболочек: Windows Console (cmd), Windows Power Shell, bash, sh, csh, zsh.

Командная оболочка представляет собой интерактивную программу. Поток команд берется со стандартного ввода. Команды shell-у пользователем передаются через командую строку. Считывать строки нужно в цикле до тех пор, пока на очередной итерации не будет получен символ конца файла (детектируется при помощи константы EOF или проверкой соответствующих возвращаемых значений функции чтения из файлов/потоков). Перед считыванием нужно вывести на экран приглашение к набору команд например, "vasia"далее символ $ и следующий за ним пробел). Длина считываемой строки ограничена размером не виртуальной памяти.

Завершение shell происходит либо по закрытию стандартного потока ввода, либо в случае выполнения встроенной команды exit (Реализованы должны быть оба варианта).

Программа должна корректно обрабатывать все ошибки, выдавая осмысленные информационные сообщения в стандартный поток ошибок. В том числе обрабатывать ошибки неправильного синтаксиса командной строки.

2 Описание языка команд shell

Командная оболочка shell выполняет группу работ, где каждая работа отделяется от другой символоми перевода строки и символом ';'. Внутри работы могут встречаться команды, их аргументы и перенаправления ввода/вывода, которые могут разделятся символами конвейера '|'.

В строке могут встречаться следующие конструкции:

- экранирование символа. Осуществляется при помощи символа \. Символ после обратного слэш не будет иметь «служебного» смысла. Например обратный слэш позволяет поставить кавычку внутри кавычек. Последовательность \\ позволяет ввести обычный одинарный слэш. Символ \ может так же наоборот означать наличие специального смысла символа в ситуации, когда без этого особого смысла небыло. Если Символ \ стоит в конце строки (следующий символ в файле перевод строки), то это означает, что перевод строки «съедается», а строка на новой строчке на самом деле является продолжением текущей.

- коментарии. При наличии во входной строке символа # (не внутри кавычек, не экранированного обратным слэшом) все символы, стоящие после решётки игнорируются.

- Подстановка значений переменных. Значение переменной – это текст, который подставляется в строку в то место, где написана конструкция `${ИМЯ_ПЕРЕМЕННОЙ}`.

3 Команды и запуск команд

Команды можно разделить на 2 группы. Первая – встроенные команды shell. Часто они не требуют запуска отдельных процессов, например команда «cd» или «pwd». Внешние команды shell – это обычные программы, которые запускаются, каждая программа в своём отдельном процессе.

Команде можно указать список её аргументов. Например:

```
1 ls -l -a ../..
```

передаст программе ls после её запуска:
в argv[0] “ls”,
в argv[1] “-l”,
в argv[2] “-a”,
в argv[3] “../..”.

После списка аргументов прогаммы допускеается указывать символы < и >, которые позволяют считать стандартный ввод из файла или вывести стандартный вывод в файл, а также последовательность символов >>, что позволяет дописать вывод программы в конец указанного файла. Например:

```
cat < file.in >> file.out
```

После перенаправлений ввода вывода допускается указать символ &, который означает запуск команды в фоновом режиме. Подробнее про фоновый режим далее.

При помощи символа | организуется конвейер. Смысл символа таков, что команда слева от символа делает свой стандартный поток вывода стандартным потоком ввода для команды справа от символа. И так продолжается дальше по цепочке, пока «вертикальные палки» не закончатся. В случае, если имело место перенаправление ввода/вывода, то приоритет отдаётся именно ему и конвейер получит себе закрытый файловый дескриптор.

Символ & допускается указывать только в конце определения исполняемой работы shell. То есть до символа задающего конвейер & указать нельзя.

4 Режим переднего плана и фоновый режим

В режиме переднего плана (foreground) работы исполняются последовательно одна за другой. Каждая работа на время своего выполнения получает терминал в свой монопольный доступ. После исполнения одной или нескольких работ (в случае, если они разделены точкой с запятой) пользователю выдаётся стандартное приглашение к вводу следующего набора команд. При этом, нажатие Ctrl+C должно приводить к посылке сигнала текущей выполняющейся работе, но не самому процессу реализующему shell. По нажатию ctrl+Z работа должна приостанавливаться, после чего выдаётся приглашение shell, которое позволяет запустить новую порцию команд или продолжить выполнение преостановленной ранее работы путём указания её номера во встроенной команде «fg» или «bg». Список имеющихся работ можно посмотреть при помощи команды «jobs».

Работа запущенная в фоновом режиме (background) не должна обращаться к терминалу. В случае обращения к терминалу процесс из работы запущенной в фоновом режиме должен быть остановлен (не завершён). Если работа запускается в фоновом режиме, то процессы не должны получать SIGINT в случае нажатия Ctrl+C на клавиатуре. В случае запуска работы в фоновом режиме shell не ждёт её завершения, а сразу либо выдаёт приглашение ко вводу следующего набора команд, либо выполняет следующую работу (например они были разделены точкой с запятой). В случае завершения фоновой работы shell информирует об этом пользователя выдавая соответствующее сообщение в стандартный поток ошибок самого shell.

5 Запуск команд из истории

Шеллы позволяют использовать короткое мнемоническое имя для запуска команды из истории запусков. Для этого используется `!100500`, здесь 100500 - это номер команды в истории команд. Конструкция должна сработать как подстановка переменной, тоесть в то место, где встретилась конструкция подствляется вводившаяся когда-то команда. Дальше могут идти конструкции с точкойзапятой, перенаправление ввода-вывода и т.п.

6 Дополнительные требования

Требуется реализовать встроенные команды:

- cd – смена текущего каталога
- pwd – выдача текущего каталога в стандартный поток вывода
- jobs, fg, bg – выдача списка активных в текущий момент работ
- exit – выход из shell
- history – команда показывающая историю команд

Требуется реализовать подстановку любых переменных, которые перед запуском выставлены в переменных окружения, в том числе служебных:

- `$цифра` – подстановка соответствующего аргумента командной строки самого shell.
- `$#` – число параметров переданное shell
- `$?` – значение статуса последнего завершившегося процесса в последней выполнившейся работе переднего плана.
- `$USER` – имя пользователя.
- `$HOME` – домашний каталог пользователя
- `$SHELL` – имя shell.
- `$UID` – идентификатор пользователя
- `$PWD` – текущий каталог
- `$PID` – pid shell

О работе с переменными окружения можно прочитать здесь: [5].

7 Рекомендации по хранению информации о работах в памяти

В рамках первого этапа необходимо научиться считывать командные строки шелл и сохранять их в оперативной памяти. Рекомендуется данные в памяти хранить как массив структур следующего вида:

```
1
struct program
3 {
    char* name;
5     int number_of_arguments;
    char** arguments;
7     char *input_file, *output_file;
    int output_type; /* 1 - rewrite, 2 - append */
9 };

11 struct job
{
13     int background;
    struct program* programs;
15     int number_of_programs;
};
```

Список литературы

[1] Справочная страница с описанием реализации shell /bin/bash: “man bash”

[2] Перевод руководства по bash скриптам `http://www.opennet.ru/docs/RUS/bash_scripting_guide`

[3] Более полное руководство по bash `http://www.gnu.org/software/bash/manual/bashref.html`

[4] Справочная страница по регулярным выражениям: “man 7 regex”

[5] Справочная страница по функции getenv “man getenv”