Вылиток А.А.

Реализация команды echo на языке Си (по мотивам K&R)

Во всех программах на языке Си, даже состоящих из нескольких файлов — единиц компиляции, определяется единая внешняя функция с именем `main`. В операционной среде UNIX имеется возможность передать аргументы или параметры запускаемой программе при помощи командной строки. Взаимодействие с операционной системой через командный интерпретатор — программой, которая воспринимает и исполняет команды (приказы) пользователя характерно для UNIX. В момент вызова программы функция `main` получает два аргумента. В первом, обычно называемом `argc` (сокращение от argument count), стоит количество аргументов, задаваемых в командной строке. Второй, `argv` (от argument vector), является указателем на массив указателей на строки, являющиеся аргументами.

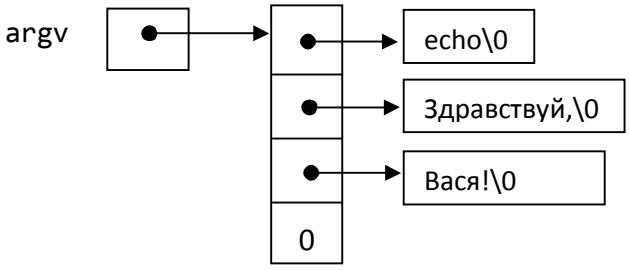
Реализуем команду echo («эхо»). Она печатает свои аргументы в одной строчке, отделяя их друг от друга пробелом. Так, команда

`$> echo Здравствуй, Вася!` напечатает

Здравствуй, Вася!

Даже если между параметрами будет несколько пробелов, в результате напечатается только один. Это связано с тем, что командный интерпретатор, формируя строки-аргументы для передачи в main, игнорирует пробельные символы.

По соглашению `argv[0]` — это имя вызываемой программы, так что значение `argc` никогда не бывает меньше 1. Если `argc` равен 1, то в командной строке после имени программы никаких аргументов нет. В нашем примере `argc` равен 3, и соответственно `argv[0]`, `argv[1]` и `argv[2]` — это строки "echo", "Здравствуй," и "Вася!". Первый необязательный аргумент — это `argv[1]`, последний — `argv[argc-1]`. Кроме того, стандарт требует, чтобы `argv[argc]` всегда был пустым указателем, т.е. его значение при приведении к целому типу равно нулю.



Первая версия программы echo трактует `argv` как массив символьных указателей.

```
#include <stdio.h>
/* эхо аргументов командной строки; версия 1 */
main(int argc, char * argv[])
{int i;
 for (i = 1; i < argc; i++)
   printf("%s%s", argv[i], (i < argc-1) ? " " : "");
 printf("\n");
```

```
 return 0;
}
```

Так как argv есть указатель на массив указателей, мы можем работать с ним как с указателем, а не как с индексируемым массивом. Следующая программа основана на продвижении argv, он продвигается так, что его значение в каждый отдельный момент ссылается на очередной указатель на char; перебор указателей заканчивается, когда исчерпан argc.

```
#include <stdio.h>
/* эхо аргументов командной строки; версия 2 */
main(int argc, char *argv [])
{
 while (--argc > 0)
     printf("%s%s", *++argv, (argc >1)?" ": "");
 printf("\n");
 return 0;
}
```

Аргумент argv — указатель на начало массива строк-аргументов. Использование в ++argv префиксной операции ++ приведет к тому, что первым будет напечатан argv[1], а не argv[0]. Каждое очередное продвижение указателя дает нам следующий аргумент, на который ссылается *argv. В это же время значение argc уменьшается на 1, и, когда оно станет нулем, все аргументы будут напечатаны.

Оператор для печати можно было бы написать и так:

```
printf((argc > 1) ? "%s " : "%s", *++argv);
```

Как видим, формат в printf может быть выражением, результатом которого является строка. Наконец, можно обойтись без использования argc, полагая что argv[argc] равно нулю, как того требует Стандарт.

```
#include<stdio.h>

/* эхо аргументов командной строки; версия 3 */

main(int argc, char* argv[])
{
    while(*++argv)
          printf( *(argv+1)?"%s " :"%s",*argv);
    printf("\n");
    return 0;
}
```

Кое-что о «настоящем» echo:

Echo the STRING(s) to standard output.

-n do not output the trailing newline

-e enable interpretation of backslash escapes

-E disable interpretation of backslash escapes (default)

--help display this help and exit

--version

output version information and exit

If -e is in effect, the following sequences are recognized:

\0NNN the character whose ASCII code is NNN (octal)

\\ backslash

\a alert (BEL)

\b backspace

\c suppress trailing newline

\f form feed

\n new line

\r carriage return

\t horizontal tab

\v vertical tab

NOTE: your shell may have its own version of echo, which usually supersedes the version described here. Please refer to your shell's documentation for details about the options it supports.