Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Факультет комп’ютерних наук

# ЛАБОРАТОРНА РОБОТА № 2

з дисципліни «Операційні системи»

Тема: «Взаємодія з середовищем виконання»

Виконав:

студент 3 курсу

групи КС-32

Безрук Юрій Русланович

Перевірив: Споров О.Є.

Харків – 2020

Целью данной работы является ознакомление с реализацией взаимодействия программ на языке С с аргументами командной строки, переданными при запуске программы, реализацией длинных и коротких опций, окружением переменных и способами получения данных о компьютере и системе.

# ХОД РАБОТЫ

**Задание 1.**

Задание 1 служит для ознакомления со способом передачи входных параметров программе при вызове в командной строке. Требуется написать программу, которая будет решать квадратное уравнение при заданных значениях коэффициентов a, b, и c. Для этого функцию main необходимо объявить как int main(int argc, char\* arhv[]), где argc – количество передаваемых агрументов (при запуске без параметров это значение равно 1), а argv – массив строк, представляющих собой передаваемые аргументы (при запуске без параметров в нулевой ячейке находится имя программы). Т.к. argv это массив строк, первым делом нужно преобразовать аргументы, преданные программе в числа, например к типу double. Сделать это можно при помощи функции atof(). Далее высчитываем дискриминант, и, если он больше нуля, высчитываем корни и выводим их на экран. В случае, если дискриминант меньше нуля, сообщаем об этом и завершаем программу. Так же добавим проверку на то, что программа запустилась с достаточным количеством аргументов, и в случае, если это не так, выводим ошибку.

**Задание 2**

Задание 2 служит для ознакомления с механизмом обработки коротких опций, переданных в массив argv. Необходимо написать программу, которой могут быть переданы короткие опции –h (помощь), -o (особое имя выходного файла), и –c (особый режим работы). Кроме того, программе могут быть переданы имена входных файлов как дополнительные аргументы. Для обработки опций стоит воспользоваться встроенной функцией int getopt(int argc, char\* argv[], const char\* optstring) (где argc и argv передаются из main, а optstring – строка, содержащая в себе символы опций, на которые функция должна обращать внимание), которая возвращает код символа текущей обрабатываемой опции, код символа ‘?’ при ошибке и -1, если анализ строки завершен. Что бы реализовать условия задачи, реализуем простой функционал для программы: она будет писать в файл информацию о его имени и режиме работы программы (обычном или особом). Файл вывода по умолчанию будет иметь имя data.txt., а режим по умолчанию – обычный. В цикле запускаем обработку функции getopt, передавая ей нужные для обработки опции: -h –c без аргументов и –o с обязательным аргументом. Далее определяем дальнейшие действия в зависимости от возвращаемого функцией значения: в случае h вызываем функцию help(), которая печатает справку по использованию программы, о – устанавливаем значение аргумента опции как новое имя выходного файла, с – изменяем режим работы на специальный. Добавляем так же обработку случаев неверной опции и недостатка аргументов. Далее записываем информацию в указанный файл. Так же добавляем обработку дополнительных аргументов: если программе при запуске было передано аргументы которые не относятся к опциям, распознавать их как имена дополнительных файлов, в которые нужно сделать запись.

**Задание 3**

Задание 3 является расширением программы задания 2 на использование длинных опций –help –output –compile . для этого функция getopt заменяется на getopt\_long(int argc, char\* argv[], const char\* optstring, const struct option\* longopts, int\* longindex), где первые три аргумента аналогичны, longopts – массив структур, в которых устанавливается взаимоотношение между короткими и длинными опциями, а longindex – указатель на переменную, в который помещается индекс обнаруженной длинной опции. Инициализируем массив структур нужными опциями. В остальном код программы идентичен.

**Задание 4**

В задании 4 необходимо ознакомиться с переменной extern char\*\* environ, которая содержит в себе массив строк, содержащих переменую окружения программы и ее значение. Требуется вывести все переменные на экран. Для этого объявляем переменную, и в цикле, получив копию адреса, выводим каждый элемент, пока не встретим NULL.

**Задание 5**

В заданиях 5-7 происходит ознакомление с функциями, предназначенными для работы с переменными окружения. Конкретно в задании 5 нужно написать программу, которая выводит в стандартный поток вывода значение переменной, переданной программе при запуске. В случае если имя переменной не указано, выводится информация об использовании программы. Таким образом, если argc = 1, печатаем справку в консоль, в противном случае для каждого из переданных аргументов выводим значение при помощи функции getenv(const char\* name).

**Задание 6**

В задании 6 требуется написать программу, которая может присваивать заданной переменной окружения переданное значение. В случае, если количество переданных аргументов меньше 3 (недостаток информации), выводим справку по программе, в противном же случае присваиваем заданной переменной заданное имя и проверяем, произошло ли присвоение, после чего выводим пару в консоль. Данная программа была реализована двумя способами (файлы set.c и put.c), при помощи функций setenv и putenv. В первом случае, функции передаются напрямую значения аргументов и флаг разрешения перезаписи переменной, а втором – значения аргументов копируются в строковые переменные, склеиваются при помощи стандартных функций работы со строками и передаются единой строкой.

**Задание 7**

Программа задания 7 должна удалять заданную при запуске переменную окружения, а в случае вызова без аргументов – удалить всё окружение. После выполнения вывести текущее окружение в стандартный поток. Таким образом, если argc равен 1, программа вызывает функцию clearenv(), которая очищает окружение, а если больше – то при помощи функции unsetenv(const char\* NAME) удаляет заданную переменную, после чего выводит список всех переменных окружения из переменной environ. В обоих случаях производятся проверки на правильность выполнения функций.

**Задание 8**

В задании 8 требуется написать программу, получающую информацию о пользователе который её запустил и о корневом пользователе. Для этого инициализируем структуру типа struct passwd\* (поля которой отображают информацию о пользователе) путём вызова функции getpwuid() которой передаём результат вызова функции getuid(), возвращающей номер пользователя. Пишем функцию вывода информации и передаем ей данную структуру. Затем переинициализируем структуру, на этом раз вызовом функции getpwnam(), которой передаём имя – «root», и так же добавляем его в фуункцию.

**Задание 9**

Задание 9 представляет из себя программу, выводящую информацию о компьютере, на котором она выполняется. Для этого выделяем память под структуру типа struct utsname\* и передаем ее функции uname(). В случае, если она не возвращает ошибку, дополнительно узнаем функцией gethostname() сетевое имя машины, а gethostid() – сетевой идентификатор. Передаём эти данные функции print\_info(), которая выводит их в наглядной форме.

**Домашнее задание №2 «Взаимодействие со средой исполнения»**

Для выполнения главного задания лабораторной работы необходимо применить навыки полученные в ходе выполнения заданий 1-7 что бы собрать полноценную программу просмотра и редактирования переменных окружения environ. Каждое действие оформлено в виде одной из функций, собранных в отдельном исходном файле, для сборки проекта подготовлен makefile. В файле opts.h размещаются подключения библиотек, необходимых для работы функций, а так же прототипы этих функций. Файл opts.c содержит непосредственную реализацию функций, а main.c – обработку и распознавание опций.

1. В случае, если программа вызывается без аргументов (argc =1), вызывается функция status(), которая выводит полный список переменных окружения с их значениями из переменной extern char\*\* environ.

2. Если указана опция -h или --help, вызывается функция help(), которая выводит подробную справку по использованию программы.

3. Если указана опция -i или --info, (с обязательным аргументом), то вызывается функция info(optarg), которая печатает значение указанной переменной если она существует.

4. Если указана опция -s или --set, (с обязательным аргументом вида переменная=значение), то вызывается функция set(optarg), которая при помощи putenv() устанавливает значение этой переменной если это возможно.

5. Если указана опция -a (или --assign) [переменная], то в специальную переменную variable записывается значение аргумента опции. Если указана опция -v (или --value) [значение] (в любом порядке), то значение записывается в специальную переменную value. Если на выходе окажется, что одна из этих переменных не равна NULL (значение по умолчанию), программа проверит, какая. Если не была задана переменная, будет выведена ошибка и обработка значения не произойдёт. Если не было задано значение, то в служебную переменную value запишется пустая строка. После этого вызовется функция assign\_value(variable, value) и создаст (или перезапишет) переменную с указанным (или пустым) значением.

6. Если указана опция -d или --del, (с обязательным аргументом), то вызывается функция del(optarg), удаляющая значение заданной переменной окружения, если это возможно.

Если указана опция -c или --clear, то вызывается функция clear(), которая очищает окружение программы при помощи функции clearenv(), если это возможно.

В случаях, когда какое-либо действие совершить невозможно, выводится соответствующее сообщение.

# ВЫВОДЫ

Таким образом, в ходе выполнения данной работы были рассмотрены работа с аргументами, опциями, окружением и данными компьютера. По завершению была разработана система environ в многофайловой структуре для управления средой окружения с широким функционалом. Скрипты сборки были помещены в makefile. Система протестирована и отлажена.