Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Факультет комп’ютерних наук

# ЛАБОРАТОРНА РОБОТА № 3

з дисципліни «Операційні системи»

Тема: «Робота з процесами в Linux»

Виконав:

студент 3 курсу

групи КС-32

Безрук Юрій Русланович

Перевірив: Споров О.Є.

Харків – 2020

Целью данной работы является ознакомление с работой процессов в операционной системе Linux и написания нескольких программ, использующих упрощенное создание процессов.

# ХОД РАБОТЫ

**Задание 2**

В задании 2 требуется реализовать выполнение консольных команд при помощи функции system() и вывести результат их выполнения в консоль. Команды подаются в виде: просто команда, команда с опиями и/или аргументами, несколько команд с опциями и аргументами, разделенных символом «;». Так как введение «;» в консоль разделит команду запуска программы со следующей команды, потребуем, чтобы пользователь в случае нескольких команд вводил их все внутри кавычек.

В общем случае, вариантов работы программы 3 (разделяем условными конструкциями):

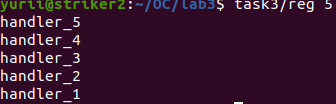
1. Программа запущена без аргументов. В этом случае выводим справку по использованию программы.
2. Программа запущена с одним аргументом. Это может быть как одна команда без аргументов, так и выражение в кавычках, содержащее множество команд, разделённых точкой с запятой. В любом случае просто передаем строку-аргумент программы функции system() и выводим результат выполнения.
3. Программа запущена с несколькими аргументами. Подразумеваем, что это команда с её опциями и аргументами, переданная без кавычек и таким образом разделенная на несколько аргументов. В этом случае, склеиваем все аргументы в единую строку, а затем передаём в функцию system(). Выводим результат завершения.

**Задание 3**

В задании 3 следует поработать с обработчиками завершения выполнения. Программе из командной строки передаётся количество обработчиков, которое нужно зарегистрировать. Сами обработчики должны задерживать выполнение на 1 секунду и выводить в консоль свой номер.

В файле handlers.h объявляем 32 идентичных функции-обработчика (32 это минимальный максимум возможного количества регистрируемых обработчиков), отличающихся лишь номером, и добавляем их адреса в массив указателей на функции, для более удобного вызова в дальнейшем. В этом же файле подключаем необходимые библиотеки. В главном файле main.c подключаем хедер.

В случае, если пользователь не передал аргументов программе, выводим справку по использованию. В противном же случае, переводим аргумент в целое число и проверяем, что он подходит выбранному диапазону (0, 32]. После чего при помощи цикла регистрируем все функции из массива. Обработчики выполняются в обратном порядке:



**Задание 4**

В задании 4 нужно создать аналог командного процессора для обработки вводимых пользователем команд с выводом результатов выполнения и обработкой завершения. Первым делом задаём размер буфера команд по умолчанию – 127 и регистрируем обработчик завершения (о нём позже). Далее добавляем обработку опций. У программы есть две коротких опции с соответствующими им длинными опциями: -h, которая выводит справку о использовании и –b, которая задаёт размер буфера команд.

Далее получаем информацию о пользователе функцией getpwuid(). Запоминаем его короткое имя и выводим остальную информацию. Далее выделяем память под буфер команд и запускаем командный процессор в цикле. На каждой итерации он выводит приглашение ко вводу команды в виде короткого имени пользователя системы. Далее при помощи функции getline() получаем команду. В случае если команда – “stop”, выходим из цикла, в противном случае передаём команду функции system() и выводим результат (или ошибку выполнения, полученную из переменной errno). После выхода из цикла программа освобождает память и начинает завершения. В этот момент вызывается обработчик завершения. Он просит пользователя подождать три секунды, ждёт три секунды при помощи функции sleep() и очищает терминал консольной командой clear. Основные моменты программы прокомментированы в коде.

**Задание 5**

Задание 5 рассчитано на вызов дочерних программ при помощи функции system(). Сама программа должна вызывать некоторое число дочерних программ, которые делают запись в файл некоторого количества случайных чисел в диапазоне [0, 1]. Имя выходного файла передается через переменную окружения FILE\_NAME. Количество дочерних программ задаётся при запуске основной программе. Количество чисел, записанных в файл зависит от номера дочерней программы. Каждая дочерняя программа генерирует свой файл вида <name><number>.txt, где name –имя файла передающееся через окружение, number – номер дочерней программы, передаваемый ей от родительской через консольный вызов.

В файле main.c находится код основной программы. Задаём количество подпрограмм по умолчанию – 1. Далее следует обработка опций, если они присутствуют. Программа имеет три опции:

1. help: выводит справку о использовании программы.
2. file (с обязательным аргументом): задаёт основную часть имени выходных файлов дочерних программ, помещая её в переменную окружения. (По умолчанию «default»)
3. number (с обязательным аргументом): задаёт количество дочерних программ которое будет создано, если оно валидно. По умолчанию – 1.

Далее программа начинает формировать строку обращения к подпрограмме, которая скомпилирована по имени sub. В цикле при помощи функции sprint() текущий номер подпрограммы переводится в строку и приклеивается к строке вызова функцией strcat(), после чего выполняется системный вызов, и изменяемая часть строки вызова очищается. Так происходит для всех дочерних программ и таким образом они все вызываются одна за дной и имеют уникальный номер.

Код дочерней программы описан в файле sub.c. Каждая дочерняя программа получает свой номер из командной строки от родительской программы. В консоль выводится о том что данный потомок начал свою работу, после чего происходит выделение памяти под имя выходного файла. По умолчанию оно задаётся как default. В случае, если из переменной окружения FILE\_NAME удалось корректно получить значение, память перевыделяется и формируется новая основная часть имени. Вне зависимости от того, было ли изменено имя, к нему добавляется номер текущей дочерней программы и расширение .txt, после чего имя и номер программы передаются функции writetofile(). В ней устанавливается зерно генератора случайных чисел, открывается на запись файл вывода по заданному имени и происходит запись количества, соответствующего номеру дочерней программы, случайных чисел в диапазоне [0,1]. При завершении дочерней программы она выводит сообщение о том что данный потом свою работу совершил.

В файле lib.h подключаются библиотеки для работы с проектом, а потом уже этот файл подключается к обоим программам для упрощения.

# ВЫВОДЫ

Таким образом, в ходе выполнения данной работы были рассмотрены упрощённые системные вызовы из программы при помощи функции system(): запуск терминальных команд и создание дочерних программ, а так же управление обработчиками завершения программы.