**ЛАБОРАТОРНАЯ РАБОТА №5 ПОТОКИ В ОС LINUX**

Цель работы – изучение потоков в ОС Linux.

**Теоретическая часть**

Существует расширенная реализация понятия ***процесс***, когда ***процесс*** представляет собой совокупность выделенных ему ресурсов и набора ***нитей исполнения***. ***Нити*** ***(threads)*** или потокипроцесса разделяют его программный код, глобальные переменные и системные ресурсы, но каждая ***нить*** имеет собственный программный счетчик, свое содержимое регистров и свой стек. Все глобальные переменные доступны в любой из дочерних нитей. Каждая нить исполнения имеет в системе уникальный номер – идентификатор ***нити***. Поскольку традиционный процесс в концепции нитей исполнения трактуется как процесс, содержащий единственную ***нить*** исполнения, мы можем узнать идентификатор этой ***нити*** и для любого обычного процесса. Для этого используется функция ***pthread\_self()***. Нить исполнения, создаваемую при рождении нового процесса, принято называть **начальной** или **главной** нитью исполнения этого процесса. Для создания нитей используется функция:

***#include <pthread.h>***

***int pthread\_create( pthread\_t \*thread, const pthread\_attr\_t \*attr,***

***void \*(\*start\_routine)( void\*), void \*arg);***

Функция создает новую нить в которой выполняется функция пользователя ***start\_routine*** ,передавая ей в качестве аргумента параметр ***arg***. Если требуется передать более одного параметра, они собираются в структуру, и передается адрес этой структуры.При удачном вызове функция ***pthread\_create*** возвращает значение ***0*** и помещает идентификатор новой нити исполнения по адресу, на который указывает параметр ***thread***. В случае ошибки возвращается положительное значение, которое определяет код ошибки, описанный в файле ***<errno.h>***. Значение системной переменной ***errno*** при этом не устанавливается. Параметр ***attr*** служит для задания различных атрибутов создаваемой нити. Функция нити должна иметь заголовок вида:

***void \* start\_routine (void \*)***

Завершение функции потока происходит если:

* функция нити вызвала функцию ***pthread\_exit()***;
* функция нити достигла точки выхода;
* нить была досрочно завершена другой нитью.

Функция ***pthread\_join()*** используется для перевода нити в состояние ожидания:

***#include <pthread.h>***

***int pthread\_join (pthread\_t thread, void \*\*status\_addr);***

Функция ***pthread\_join()*** блокирует работу вызвавшей ее нити исполнения до завершения нити с идентификатором ***thread***. После разблокирования в указатель, расположенный по адресу ***status\_addr***, заносится адрес, который вернул завершившийся ***thread*** либо при выходе из ассоциированной с ним функции, либо при выполнении функции ***pthread\_exit()***. Если нас не интересует, что вернула нам нить исполнения, в качестве этого параметра можно использовать значение ***NULL***.

Для компиляции программы с нитями необходимо подключить библиотеку ***pthread.lib*** следующим способом:

***gcc 1.c –o 1.exe -lpthread***

**Порядок выполнения работы**

1. Изучить теоретическую часть лабораторной работы.
2. Написать программу, создающую два дочерних потока. Родительский процесс и два дочерних потока должны выводить на экран свой ***id*** и ***pid*** родительского процесса и текущее время в формате: ***часы: минуты: секунды: миллисекунды***.

Варианты индивидуальных заданий

1. Написать программу шифрования всех файлов для заданного каталога его подкаталогов. Пользователь задаёт имя каталога. Главный процесс открывает каталоги и запускает для каждого файла каталога отдельный поток шифрования файла. Каждый поток выводит на экран свой ***id,*** полный путь к файлу, общее число зашифрованных байт. Число одновременно работающих потоков ***N*** (вводится пользователем). Проверить работу программы для каталога ***/etc***.
2. Написать программу нахождения массива ***N*** последовательных значений функции ***y[i]=sin(2\*PI\*i/N)*** ***( i=0,1,2…N-1 )*** с использованием ряда Тейлора. Пользователь задаёт значения ***N*** и количество ***n*** членов ряда Тейлора. Для расчета каждого члена ряда Тейлора запускается отдельный поток. Каждый поток выводит на экран и в файл свой ***id*** и рассчитанное значение ряда. Головной процесс считывает из файла значения всех рассчитанных членов ряда Тейлора, суммирует их и полученное значение ***y[i]*** записывает в файл. Проверить работу программы для ***N=256 n=5; N=1024 n=10.***
3. Написать программу синхронизации двух каталогов, например, ***Dir1*** и ***Dir2***. Пользователь задаёт имена ***Dir1*** и ***Dir2***. В результате работы программы файлы, имеющиеся в ***Dir1***, но отсутствующие в ***Dir2***, должны скопироваться в ***Dir2*** вместе с правами доступа. Процедуры копирования должны запускаться в отдельном потоке для каждого копируемого файла. Каждый поток выводит на экран свой ***id,*** полный путь к копируемому файлу и число скопированных байт. Число одновременно работающих потоков ***N*** (вводится пользователем). Скопировать несколько файлов из каталога ***/etc*** в свой домашний каталог. Проверить работу программы для каталога ***/etc и*** домашнего каталога.
4. Написать программу поиска одинаковых по содержимому файлов в двух каталогов, например, ***Dir1*** и ***Dir2***. Пользователь задаёт имена ***Dir1*** и ***Dir2***. В результате работы программы файлы, имеющиеся в ***Dir1***, сравниваются с файлами в ***Dir2*** по их содержимому. Процедуры сравнения должны запускаться в отдельном потоке для каждой пары сравниваемых файлов. Каждый поток выводит на экран свой ***id***, имя файла, общее число просмотренных байт и результаты сравнения. Число одновременно работающих потоков ***N*** (вводится пользователем). Скопировать несколько файлов из каталога ***/etc*** в свой домашний каталог. Проверить работу программы для каталога ***/etc и*** домашнего каталога.
5. Написать программу поиска заданной пользователем комбинации из ***m*** байт (***m <255***) во всех файлах текущего каталога. Пользователь задаёт имя каталога. Главный процесс открывает каталог и запускает для каждого файла каталога отдельный поток поиска заданной комбинации из ***m*** байт. Каждый поток выводит на экран свой ***id,*** полный путь к файлу, общее число просмотренных байт и результаты (сколько раз найдена комбинация) поиска. Число одновременно работающих потоков ***N*** (вводится пользователем). Проверить работу программы для каталога ***/etc и*** строки ***ifconfig***.
6. Написать программу подсчета количества слов в файлах заданного каталога его подкаталогов. Пользователь задаёт имя каталога. Главный процесс открывает каталоги и запускает для каждого файла каталога отдельный поток подсчета количества слов. Каждый поток выводит на экран свой ***id,*** полный путь к файлу, общее число просмотренных байт и количество слов. Число одновременно работающих потоков ***N*** (вводится пользователем). Проверить работу программы для каталога ***/etc и*** строки ***ifconfig***.
7. Написать программу подсчета частоты встречающихся символов в файлах заданного каталога его подкаталогов. Пользователь задаёт имя каталога. Главный процесс открывает каталоги и запускает для каждого файла каталога и отдельный поток подсчета частоты встречающихся символов. Каждый поток выводит на экран свой ***id,*** полный путь к файлу, общее число просмотренных байт и количество слов. Число одновременно работающих потоков ***N*** (вводится пользователем). Проверить работу программы для каталога ***/etc и*** строки ***ifconfig***.
8. Поиск плагиата. Имеются два файла (файл1 и файл2) с текстами. Необходимо найти совпадающие по содержанию куски текста вывести их в файл результата с указанием номера начальной позиции в файле 1 и в файле2 и размера куска текста. Для ускорения работы использовать потоки. Число одновременно работающих потоков N (вводится пользователем).