Лабораторная работа № U9

Создание процессов и потоков

Процесс – программа, находящаяся на исполнении, при этом ей предоставляется процессорное время.

Поток – иногда называется «легковесный процесс» или «независимый поток управления в рамках процесса». В рамках одного процесса может быть запущено несколько потоков, у них у всех будет одинаковый PID (идентификатор процесса) и эти потоки могут работать с любыми адресами в рамках виртуального адресного пространства потока.

Для создания нового процесса в Unix используется системный вызов **fork**.

Пример команд для создания и управления потоками Unix:

- **pthread_create** – запуск потока

*int pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine) (void *), void *arg)*;

- первый параметр (*pthread_t *thread*)– указатель на структуру в которую будут сохранены сведения о созданном потоке
- второй параметр (*const pthread_attr_t *attr*) указатель на структуру с параметрами создаваемого потока, можно не указывать (значение NULL)
- третий параметр (*void *(*start_routine) (void *)*) – функция, которая будет запускаться как поток
- четвертый параметр (*void *arg*) – аргумент, передаваемый в функцию

При этом функция, которая будет запускаться как поток должна быть определена как:

Void * thread_function_name(void *arg)

Где thread_function_name – название функции, может быть любым, а arg – указатель на параметр, передаваемый при запуске потока, например, для извлечения переданного числа может быть использована такая конструкция myint = * (int *) arg;

- **pthread_join** – подключиться к потоку и ожидание его завершения

int pthread_join(pthread_t thread, void **retval);

***Примечание**: 1) Для работы с потоками необходимо подключить библиотеку pthread.h и при компилировании (с помощью gcc) указывать компилятору необходимость использования библиотеки pthread, с помощью опции –pthread . Например так: gcc –o program –pthread prog.c*

2) для sleep нужна библиотека unistd.h

Задание:

1. Написать программу N1 – которая будет:
   a. Печатающую на экран свой идентификатор
   b. создающую новый процесс и выполняющую следующие действия после того как будет выполнен fork:
      i. Потомок на экран должен вывести текст типа «I am child» и свой идентификатор (getpid). После этого перейти в ожидание (sleep) на 10 секунд и завершиться

ii. Предок на экран должен вывести текст «I am parent» и свой идентификатор (getpid) и перейти в ожидание завершения потомка (wait). После того как потомок завершиться, вывести на экран какой-либо текст и завершиться.

2. Запустить программу и в ещё одной копии терминале по списку процессов:
   a. найти запущенные процессы (ps –la), обратить внимание на колонки PID (ProcessID, идентификатор процесса) и PPID (Parent PID, идентификатор процесса-предка)
   b. определить используется ли процессор на обе копии программы (командой top)
3. завершить командой kill процесс потомок и убедиться, что предок также завершился
4. снова запустить программу:
   a. найти запущенные процессы (ps –la), определить какие значения PID и PPID
   b. командой kill завершить предка
   c. с помощью команды ps –la вывести список процессов, проверить завершился ли потомок и посмотреть значения его PID и PPID
5. написать программу N2 - которая будет запускать два потока:
   a. алгоритм потоков – определён должен быть один раз. При запуске потока на экран должно выводиться сообщение о значении входного параметра и значение идентификатора процесса (PID), внутри потока должен быть цикл, количество итераций которого зависит от входного параметра, внутри цикла:
      i. печать на экран значения входного параметра и номера текущей итерации
      ii. выполнение системного вызова для паузы в 3 секунды– sleep(3)

      по завершению цикла напечатать на экран сообщение о завершении, с указанием значения входного параметра
   b. через входной параметр передаётся число 5 для первого, и 10 для второго потока
   c. pthread_join должен ждать завершения потока N2, до и после pthread_join реализовать вывод на экран «отладочного» сообщения.
   d. в самом начале и в самом конце main вывести на экран сообщение о завершении программы с указанием её идентификатора (PID)
6. откомпилировать и запустить программу – *определить* – отработали ли потоки запрограммированную логику или нет
7. создать копию исходного кода программы и модифицировать уже копию:
   pthread_join должен ждать завершения потока N1 [прим. ранее был поток N2]
8. откомпилировать и запустить программу – *определить* – отработали ли потоки запрограммированную логику или нет
9. создать копию исходного кода программы и модифицировать уже копию:
   убрать pthread_join.
10. откомпилировать и запустить программу – *определить* – отработали ли потоки запрограммированную логику или нет