Lab 1

Как создать поток?

Традиционная модель процессов в UNIX поддерживает только один поток управления на процесс. Концептуально это то же, что и модель, основанная на потоках, в случае, когда каждый процесс состоит из одного потока. При наличии поддержки pthread программа также запускается как процесс, состоящий из одного потока управления. Поведение такой программы ничем не отличается от поведения традиционного процесса, пока она не создаст дополнительные потоки управления. Создание дополнительных потоков производится с помощью функции pthread_create.

```
SYNOPSIS
       #include <pthread.h>

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

       Compile and link with -pthread.
```

```
ERRORS
       EAGAIN Insufficient resources to create another thread.

       EAGAIN A system-imposed limit on the number of threads was encountered.
              There  are  a  number of limits that may trigger this error: the
              RLIMIT_NPROC soft resource limit (set via  setrlimit(2)),  which
              limits  the  number of processes and threads for a real user ID,
              was reached; the kernel's system-wide limit  on  the  number  of
              processes and threads, /proc/sys/kernel/threads-max, was reached
              (see proc(5)); or the maximum  number  of  PIDs,  /proc/sys/ker-
              nel/pid_max, was reached (see proc(5)).

       EINVAL Invalid settings in attr.

       EPERM  No permission to set the scheduling policy and parameters speci-
              fied in attr.
```

Вновь созданный поток начинает выполнение с функции start_routine . Эта функция принимает единственный аргумент arg — нетипизированный указатель. Если функции start_routine()

потребуется передать значительный объем информации, ее следует сохранить в виде структуры и передать в arg указатель на структуру. При создании нового потока нельзя заранее предположить, кто первым получит управление — вновь созданный поток или поток, вызвавший функцию pthread_create . Новый поток имеет доступ к адресному пространству процесса и наследует от вызывающего потока среду окружения арифметического сопроцессора и маску сигналов, однако набор сигналов, ожидающих обработки, для нового потока очищается.

Завершение процесса системным вызовом `exit(2)` или возвратом из функции `main` приводит к завершению всех нитей процесса. Это поведение описано в стандарте POSIX, поэтому ОС, которые ведут себя иначе (например, старые версии Linux), не соответствуют стандарту. Если вы хотите, чтобы нити вашей программы продолжали исполнение после завершения `main`, следует завершать `main` при помощи вызова `pthread_exit(3C)`.