

Лекция 32

Нити (threads)

Нити

- Нить (легковесный процесс) — единица планирования времени ЦП в рамках одного процесса
- Все нити разделяют общее адресное пространство, открытые файловые дескрипторы и т. д.
- Каждая нить имеет свой стек, свой набор регистров

Главная нить

- При создании процесса создается главная (основная) нить. В рамках основной нити вызывается функция `main`.
- Процесс завершается, когда завершается главная нить, либо выполняется системный вызов `exit()`, либо процесс получает сигнал, вызывающий завершение процесса

Уровни планирования нитей

- 1:1 — каждая нить процесса планируется к выполнению на уровне ядра (ядро хранит информацию о всех нитях процесса)
- N:1 — все нити процесса планируются к выполнению на уровне процесса (ядро не имеет информации о нитях процесса)
- N:M — гибридный подход, используется и планирование на уровне процесса, и планирование на уровне ядра

Уровни планирования нитей

- Планирование на уровне процесса — меньше накладные расходы (нет переключения в режим ядра), но необходимо специальным образом работать с системными вызовами, которые могут заблокировать процесс, так как будут заблокированы все нити
- Планирование на уровне ядра — возрастает нагрузка на планировщик процессов, но отсутствуют проблемы с блокирующими вызовами. Кроме того, ядро может запускать нити параллельно на нескольких ядрах или процессорах.

POSIX threads (pthread)

- Стандарт на интерфейс работы с нитями
- Реализован во всех UNIX-системах
-

C11 threads

- Необязательны в C11
- Заголовочный файл: `<threads.h>`
- Не реализованы в стандартной библиотеке Си на Linux (glibc), существуют альтернативные реализации
- Неуловимый Джо?

C++ threads

- Появились в стандарте C++11
- Заголовочный файл `<thread>`
- Удобнее в использовании, чем `pthread`
- Cross-platform, но меньше возможностей, чем `pthread`
- Содержат абстракции асинхронного исполнения `<future>`

POSIX threads (pthread)

- Прототипы функций и типов данных объявлены в файле `<pthread.h>`
- Функции и типы данных имеют префикс `pthread_`
- Для компиляции программы с помощью gcc используется ключ `-pthread`

```
gcc -Wall -g -pthread prog.c -o prog
```

Главная функция нити

- Главная функция запускается при старте нити
- Когда главная функция завершает работу, нить уничтожается

```
void *start_routine(void *arg);
```

- Параметр `arg` позволяет передавать любой указатель в нить
- Возвращаемое значение передается в нить, которая ожидает завершения данной нити

Создание нити

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

- `thread` — переменная для сохранения идентификатора нити
- `attr` — дополнительные атрибуты создаваемой нити
- `start_routine` — главная функция
- `arg` — параметр, передаваемый в главную функцию

Завершение нити

```
void pthread_exit(void *value_ptr);
```

- Завершение работы нити, `value_ptr` — возвращаемое значение
- Если нить выполняет недопустимую операцию (то, что на уровне процесса вызвало бы посылку сигнала SIGSEGV или подобного ядром), завершается весь процесс

Ожидание завершения нити

```
int pthread_join(pthread_t thr, void **vptr);
```

- thr — идентификатор нити, завершение которой ожидается
- vptr — адрес переменной, в которую будет записан указатель, возвращенный главной функцией нити
- Если нить thr выполняется, текущая нить будет приостановлена до окончания работы нити thr
- Нить можно ждать только один раз
- Для созданной нити либо должна быть выполнена операция join, либо нить должна быть объявлена фоновой

Атрибуты нити

- Joinable/detached — detached нити не требуют (и не могут) ожидания завершения (join) — фоновые нити
- Processor affinity — ядра, к которым привязывается нить
- Тип планирования, параметры планирования
- Размер стека

pthread_attr_t

- Перед использованием должен быть проинициализирован
`int pthread_attr_init(pthread_attr_t *attr);`
- После использования должен быть очищен:
`int pthread_attr_destroy(pthread_attr_t *attr);`
- Специальные функции позволяют модифицировать атрибуты нити
- Затем структура атрибутов передается в `pthread_create`

Detach state

```
int pthread_attr_setdetachstate(pthread_attr_t  
*attr, int detachstate);
```

- detachstate:
 - PTHREAD_CREATE_DETACHED
 - PTHREAD_CREATE_JOINABLE

Размер стека

- По умолчанию берется из ограничений `ulimit` процесса (обычно — 8 MiB)
- Стек нитей, в отличие от стека процесса, выделяется сразу и автоматически не расширяется
- Внизу стека находится специальная «guard page»

```
int pthread_attr_setstacksize(pthread_attr_t *attr,  
size_t stacksize);
```

Processor affinity

- Маска процессоров (ядер), на которых можно выполнять нить (или процесс)
- Оптимизированным размещением нитей по ядрам можно повысить производительность

```
int pthread_attr_setaffinity_np(pthread_attr_t *attr,  
    size_t cpusetsize,  
    const cpu_set_t *cpuset);
```

- Работа с `cpu_set_t` аналогична `fd_set`, см. `man 3 cpu_set`

Нити и процессы

- Fork() можно вызывать в нитях, но
 - Fork() копирует ВСЕ адресное пространство (в режим copy-on-write)
 - В сыне продолжит выполняться только одна нить — та, которая выполнила fork
 - Остальные нити не продолжат выполнение, несмотря на то, что их состояние тоже было скопировано
- Потенциальные проблемы с мьютексами и сигнальными переменными!
- Бороться с проблемами: pthread_atfork

Нити и `exec`

- `Exec` полностью замещает адресное пространство процесса
- Все прочие нити, которые выполнялись во время `exec`, убиваются
- Состояние разделяемых ресурсов может остаться нецелостным
- Комбинация `fork/exec` относительно безопасна (но см. замечания к `fork`)

Нити и сигналы

- Сигналы приходят в процессы, а не в нити, послать сигнал одной нити нельзя
- Обработчик сигнала один на процесс
- Если поступает сигнал, то обработчик вызывается в контексте КАКОЙ-НИБУДЬ нити, в которой эти сигналы не заблокированы

Нити и сигналы

- Каждая нить может блокировать сигналы по-своему:

```
int pthread_sigmask(int how, const sigset_t *set,  
    sigset_t *oldset);
```

- Если требуется обрабатывать сигналы, то во всех нитях, кроме одной, все сигналы должны быть заблокированы

Принудительное завершение нитей

- Как правило, нити следует завершать только «добровольно»

```
int pthread_cancel(pthread_t thread);
```

- Выполняет запрос на завершение нити
 - По умолчанию нить может быть завершена только в т. н. cancellation points (обычно — системные вызовы)
 - Может быть включен «асинхронный режим», по которому нить будет завершена немедленно

```
int pthread_setcancelstate(int state, int *oldstate);
```

```
int pthread_setcanceltype(int type, int *oldtype);
```

Thread-Local Storage (TLS)

- По умолчанию все глобальные переменные являются общими для всех нитей
- Каждая нить имеет свой стек, но другие нити имеют доступ в стек нити (например, если был передан адрес)
- Ключевое слово `_Thread_local` (C11) или `thread_local` (C++) для обозначения TLS, например:

```
_Thread_local volatile int count = 5;
```


TLS

- При создании нити выделяется память под TLS
- Начальные значения переменных берутся те, которые были заданы при инициализации
- Каждая нить работает со своей копией TLS
- Все TLS находятся в общем адресном пространстве
- В реализации Linux используется сегментный регистр `gs`