Операционные системы и сети

Многопоточное программирование

Павел Филонов 06/02/2021

Программный поток (thread)

- Программным потоком (легковесным процессом) называются потоки исполнения в рамках одного адресного пространства процесса.
- Каждый процесс имеет минимум один программный поток

Процесс (совместно используемые поля всеми потоками)	**Программный поток (индивидуально для каждого потока)**
• Адресное пространство • Глобальные переменные • Открытые файлы • Дочерние процессы • Сигналы и их обработчики • Учетная информация	• Счетчик команд • Регистры • Стек • Состояние

POSIX Threads

pthreads

- Интерфейс работы с потоками описанный в POSIX
- Включает в себя как интерфейсы для управления потоками, так и для их синхронизации
- Внутри использует специфичные для ОС системные вызовы
  - Например на Linux для создания потоков используется вызов clone

Пример. Создание потока

man pthread_create

PTHREAD_CREATE(3) BSD Library Functions Manual PTHREAD_CREATE(3)

NAME
pthread_create -- create a new thread

SYNOPSIS
#include <pthread.h>

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

RETURN VALUES
If successful, the pthread_create() function will return zero. Otherwise an error number will be returned to indicate the error.

Состояние гонки

Race conditions

- Состояние гонки - ошибка проектирования многопоточного приложения, при котором работы приложения зависит от порядка выполнения части кода.
- Сложно диагностируемая ошибка, которая может случайным образом либо проявляясь, либо нет при разных запусках приложения
- Для избежания необходимо знать типовые ошибки и методы их предотвращения

Пример. Продажа билетов

- Обнаружение ситуации гонок при помощи Thread sanitizer

Взаимное исключение

Mutual Exclusion (mutex)

- **Мьютекс** - примитив синхронизации, обеспечивающий взаимное исключение исполнения *критических* участков кода
- **Критический** участок кода - участок кода, в котором производится доступ к общему ресурсу (данным или устройству), который не должен быть одновременно использован более чем одним программным потоком.

Пример. Мьютекс

Man pthread_mutex_lock

PTHREAD_MUTEX_LOCK(3) BSD Library Functions Manual PTHREAD_MUTEX_LOCK(3)

NAME

pthread_mutex_lock -- lock a mutex

SYNOPSIS

```
#include <pthread.h>

int
pthread_mutex_lock(pthread_mutex_t *mutex);
```

DESCRIPTION

The pthread_mutex_lock() function locks mutex. If the mutex is already locked, the calling thread will block until the mutex becomes available.

RETURN VALUES

If successful, pthread_mutex_lock() will return zero, otherwise an error number will be returned to indicate the error.

SEE ALSO

pthread_mutex_destroy(3), pthread_mutex_init(3), pthread_mutex_trylock(3), pthread_mutex_unlock(3)

Взаимная блокировка

deadlock

- Взаимная блокировка - ошибка проектирования многопоточного приложения, при которой несколько потоков находятся в ожидании ресурсов, занятых друг другом, и ни один из низ не может продолжать выполнение.
- Пусть имеется 2 ресурса (мьютекса) A и B

Поток 1	**Поток 2**
Хочет захватить A и B. Начинает с A	Хочет захватить A и B. Начинает с B
Захватывает A	Захватывает B
Ожидает освобождения B	Ожидает освобождения A
Взаимная блокировка (deadlock)	

Пример. Взаимная блокировка

Deadlock

- Один из способов избежать взаимной блокировки - использовать однозначный в каждом потоке порядок захвата мьютексов

Задача об обедающих философах

- Каждый философ может либо есть, либо размышлять
- Философ может есть только когда держит в руках 2 вилки
- Можно взять или положить только ближайшие 2 вилки (справа или слева)
- Нужно разработать алгоритм, при котором философы не будут голодать