Многопоточное программирование на C++

Файлы, процессы и потоки

Процессы

- Исполняемый код
- Ресурсы
  - Виртуальная память
  - Виртуальный процессор
  - Файлы
  - Пространства имен

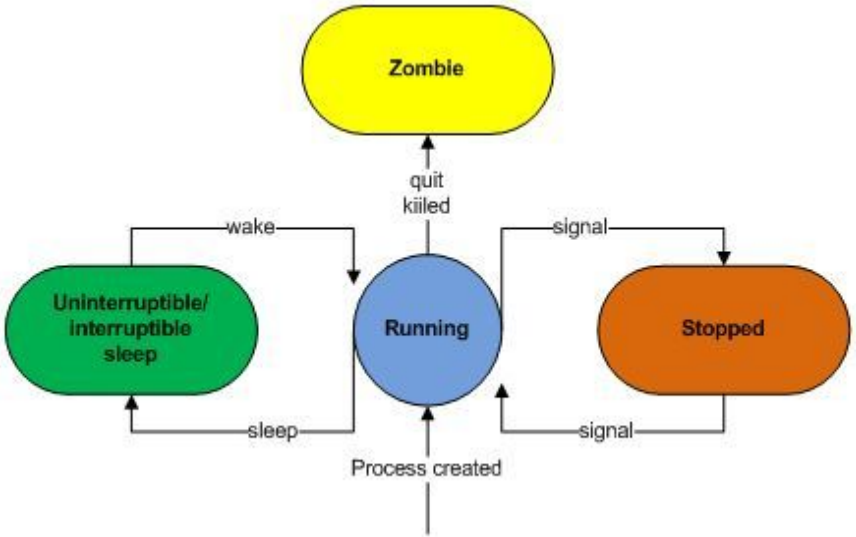
/include/linux/sched.h

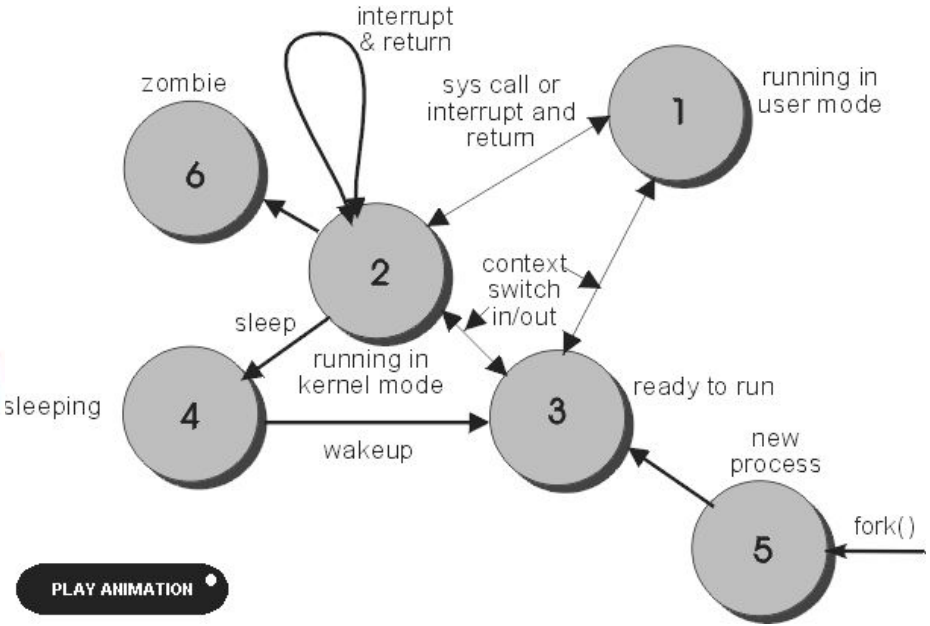
```
struct task_struct {
    // … many more

    usigned long state;
    int prio;
    unsigned long policy;
    struct task_struct *parent;
    struct list_head tasks;
    pid_t pid

    // … many more
}
```

Жизненный цикл процесса





Управление процессами

- sys/types.h, unistd.h
  - **fork()** - клонирует текущий процесс
    - == 0, дочерний процесс
    - > 0, родительский процесс
    - < 0, ошибка
  - **execl() and execlp():**
    - Наследуют переменные окружения
  - **execv() and execvp():**
    - Так же как и execl, только параметры передаются по-другому
  - **execve()**
    - Задает параметры окружения
  - **wait()**
    - Ожидание очерних процессов или сигналов

Потоки

- С точки зрения Linux такой же процесс, как и другие
- Разделяют некоторые ресурсы с родителем

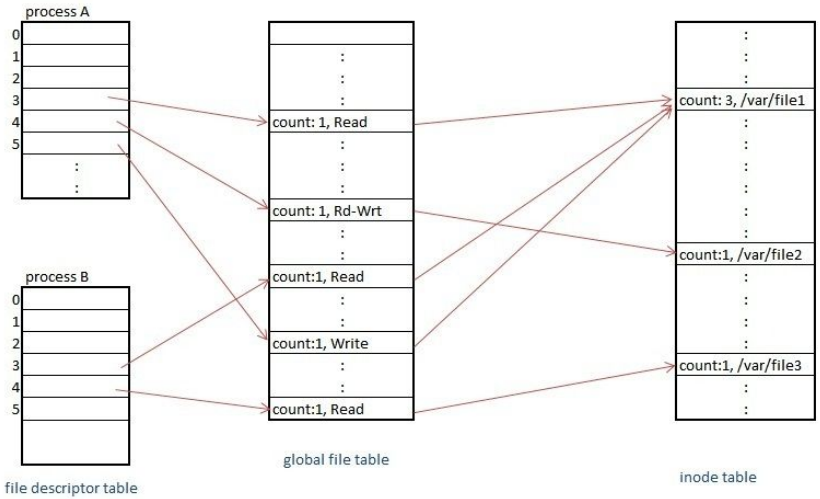
```
clone(
  CLONE_VM |
  CLONE_FS |
  CLONE_FILES |
  CLONE_SIGHAND,
  0
)
```

Управление потоками

- pthreads (POSIX threads)
  - **LinuxThreads** (not supported since libc 2.4)
  - **NPTL** (since glibc 2.3.2 and linux 2.6)
  - **pthread_create() -** Создает новый поток
  - **pthread_exit() -** Останавливает текущий поток
  - **pthread_join() -** Ждет остановки указанного потока
  - <много других методов, man pthread>...

Файлы

- POSIX
  - **open()** - открывает файл
  - **close()** - закрывает файл
  - **read()** - читает из файла
  - **write()** - пишет в файл
  - **<а так же много других…>**
  - **<как обычно man ….>**
- **unistd.h**
  - **STDIN_FILENO** - ввод
  - **STDOUT_FILENO** - вывод
  - **STDERR_FILENO** - ошибки
- **fnctl….**



Показ “кодов”