Межпроцессное взаимодействие: каналы

Лекция 15
АКОС 2019-2020

Процесс в UNIX (и не только)

- Изолированное адресное пространство
- Виртуальность адресного пространства гарантируется процессором

- Побочный эффект: требуются дополнительные действия для взаимодействия разных процессов

Способы взаимодействия

- Сигналы (про них уже рассказывалось)
  - Обычные сигналы UNIX
  - POSIX Real-Time Extensions
- Разделяемые страницы памяти: mmap(..., MAP_SHARED, ...)
- Обычные файлы

Каналы в UNIX

```
          ls          -l          |          wc
#   команда     аргумент   палочка   команда
```

- Запускается две команды
- Стандартный поток вывода первой команды перенаправляется на стандартный поток ввода второй команды
- Используемый при этом механизм - канал (pipe)

Канал в UNIX

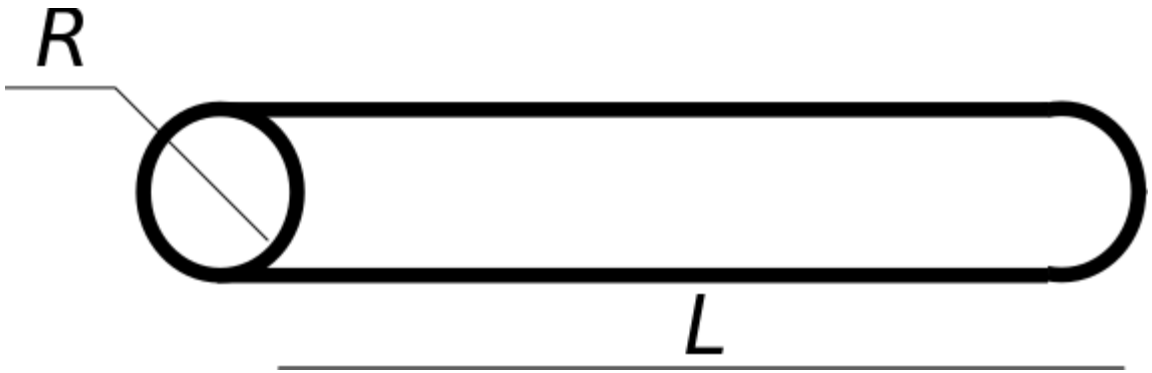
- Системный вызов `pipe(int fds[2])`
  - параметр - куда записать пару дескрипторов
  - индекс 0 - для чтения, индекс 1 - для записи

Использование каналов

- Межпроцессное взаимодействие:
  - процессы наследуют открытые файловые дескрипторы
  - после создания канала можно делать fork+exec
- В пределах одного процесса:
  - однопоточность - пока есть место
  - использовать разными потоками выполнения

Демонстрация: pipe-one-process.c

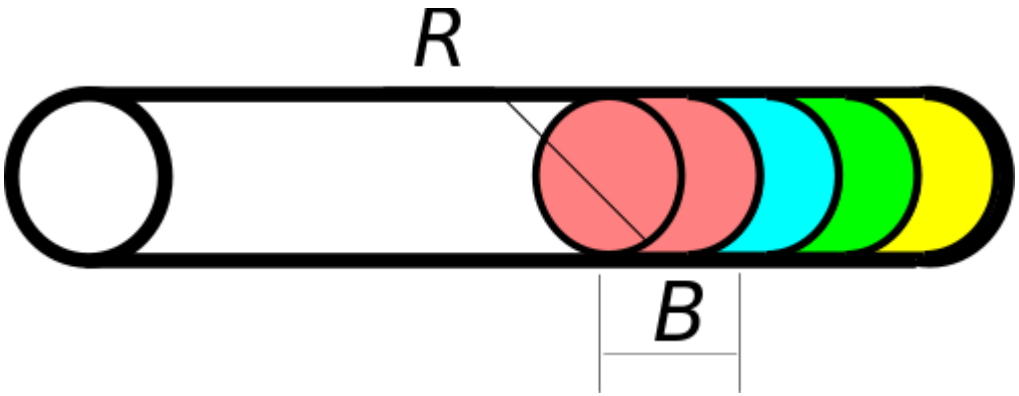
Канал в UNIX



$$V = \pi R^2 \cdot L = 65356 \text{ байт}$$

(в старых версиях Linux: размер страницы памяти)

Канал в UNIX



$$Q = \pi R^2 \cdot B = \text{PIPE\_BUF байт}$$

- Гарантируется **атомарность** чтения/записи блоков размером `PIPE_BUF`
- Для Linux (x86) размер `PIPE_BUF = 4096`
- По стандарту POSIX.1-2001 размер `PIPE_BUF >= 512`

Когда возникает проблема атомарности

- Несколько потоков в пределах одного процесса разделяют один канал

- Несколько процессов имеют файловые дескрипторы, связанные с одним каналом

Запись в канал

- Если в канале есть место и его противоположная часть открыта **хотя бы одним** процессом - происходит запись
- Если нет места:
  - запись блокируется до тех пор, пока кто-то не прочитает; либо
  - если установлен флаг O_NONBLOCK, то write завершается с ошибкой EAGAIN
- Если канал **никем не открыт на чтение**, то получаем сигнал SIGPIPE и write завершается с ошибкой EPIPE

Демонстрация: pipe-one-process.c сразными модификациями

Неблокирующий ввод-вывод

- При открытии файлов можно указать флаг `O_NONBLOCK`
- Чтение и запись из/в неблокирующие дескрипторы не приводят к остановке, а возвращают ошибку `EAGAIN`
- Для уже созданных файловых дескрипторов можно использовать `fcntl`

Чтение из канала

- Если есть данные - читается всё, что есть
- Если пустой буфер и противоположная сторона открыта хотя бы в одном процессе - ожидание данных (кроме O_NONBLOCK)
- Если закрыты все файловые дескрипторы для записи - признак конца файла

Dead Lock

- Ситуация, когда пытаемся читать что-то из канала, в который никто не собирается писать
- При запуске процессов обычно проявляется из-за забытого `close` у родителя

Демонстрация: pipe-dead-lock.c

Каналы в UNIX

```
          ls          -l          |          wc
#   команда     аргумент   палочка   команда
```

- 0 и 1 - стандартные потоки ввода и вывода
- Что нам создаст pipe - иногда предсказуемо, но в общем случае - random

dup

- dup - создание копии ФД:
  - `int dup(int oldfd);`
  - какое-то совсем legacy
- **dup2 - создание копии ФД с номером**
- dup3 - создание копии ФД с номером и флагами; Linux only

Копия ФД - очень низкоуровневый аналог Shared Pointer в C++

dup v.s. dup2

```
int dup2(int oldfd, int newfd);
```

1. Закрыть newfd, если он открыт
2. newfd = dup(oldfd)

Две операции происходят атомарно!

Именованные каналы

- Обычные каналы могут связывать только **родственные** процессы
- Пример:
  `ls -l | wc`
  Процесс bash создает канал, затем запускает два дочерних, которые наследуют эти файловые дескрипторы
- Можно договориться о том, как найти канал в неродственных процессах: по имени

Именованные каналы

- FIFO - специальный тип файла
- Создается *функцией* `mkfifo` (оболочка поверх `mknod`) или командой `mkfifo`

- Для превращения в файловый дескриптор - обычная операция открытия файла
- Ведет себя как неименованный канал

Демонстрация: 1) mkfifo channel && strace cat channel; 2) echo hello >channel

Каналы: в реальной жизни

- Файловые дескрипторы 0, 1 и 2
- Использование сторонних программ как компонент

- Типовой пример: отладчик gdb
  .... а еще gcc

- Обычно используются неименованные каналы (pipe), а не FIFO

Ограничения каналов для IPC

- Однонаправленность
- Проблемы с атомарностью, если несколько читателей/писателей

Решение: сокеты

- Двунаправленный обмен данными
- Можно отслеживать события
- Унифицированный программный интерфейс для IPC и сетевого взаимодействия

- Для настройки нужно больше букв в коде

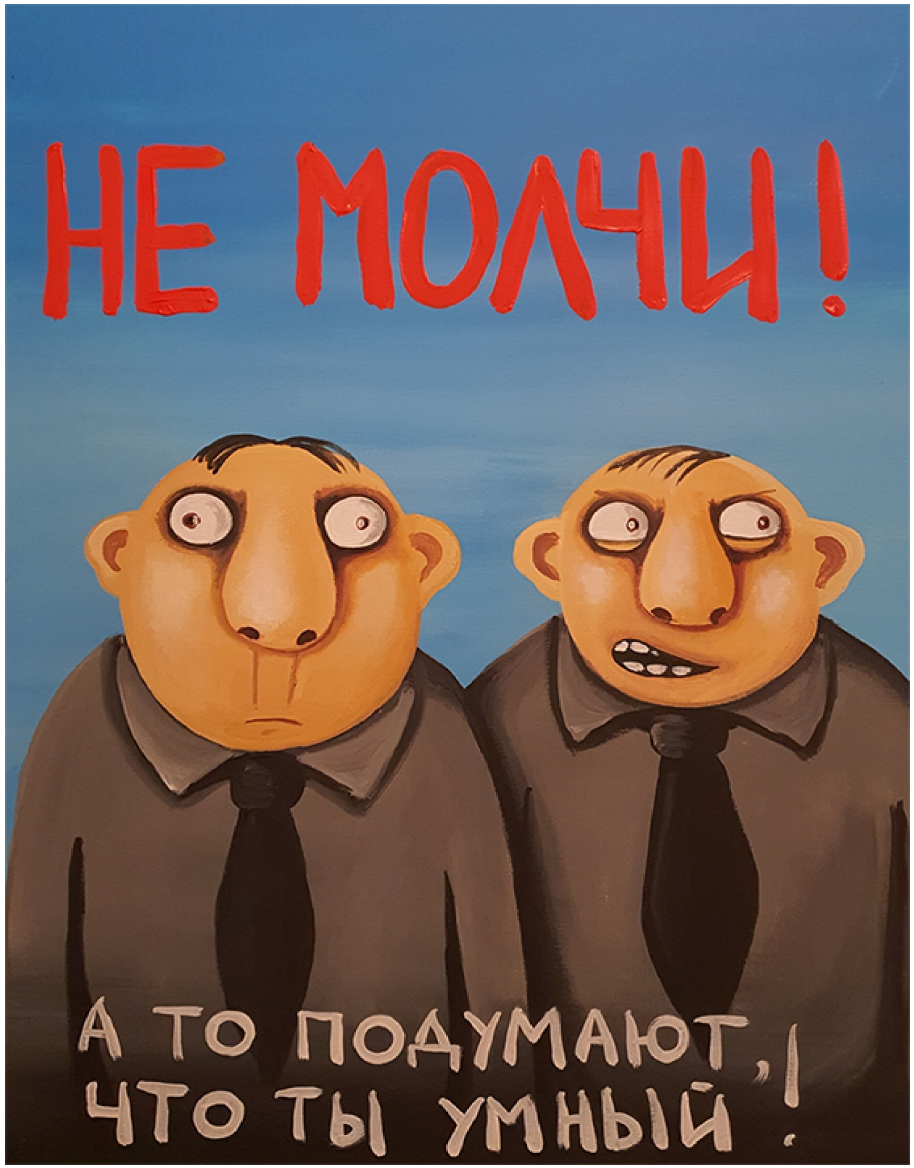
Примитив: socket pair

- Только FreeBSD и Linux
- Двунаправленный канал связи

```
socketpair(AF_LOCAL,
           SOCK_STREAM,
           0,
           out: int fds[2]
           ) → err;
```

Краткое введение в сокеты

- Краткое, потому что этому посвящена следующая лекция
- Сокеты бывают разных видов, но используются только два из них:
  - локальные (AF_LOCAL)
  - сетевые (AF_INET или AF_INET6)



Лекция - всё.

Вопросы?

Пожелания?

Немного про семестр

- 80% - домашки
- 20% - контрольная

Дедлайны

- Мягкий дедлайн - 2 недели с открытия
- Жесткий дедлайн - 4 недели с открытия
- Ещё один жесткий дедлайн - конец зачетной недели

Пересдачи!

- Кто продолбался в осеннем семестре - сдавайте домашки, зачтется в первую попытку (если не скатали, то сможете защитить).
- Кто опять продолбается - комиссия. На комиссии ничего не учитывается, начинаете с чистого листа. Но нужно будет решить задачку за 1 астрономический час + ответить теорию.