Разделяемая память. Семафоры

АКОС №16

Взаимодействие процессов

- Через файловый ввод-вывод
- Каналы (pipes)
- Именованные каналы (FIFO)
- Сигналы реального времени
- **Разделяемые страницы памяти (mmap)**
- Сокеты - позволяют одному процессу общаться сразу со многими процессами

Каналы (pipe и FIFO)

- Пара файловых дескрипторов
- Гарантируется последовательность доставки данных

Каналы (pipe и FIFO)

Чтение

```
char buf[BUF_SIZE];
ssize_t cnt;
while (
(cnt=read(fd,buf,sizeof(buf)
) > 0)
{
    ....
}
```

Запись

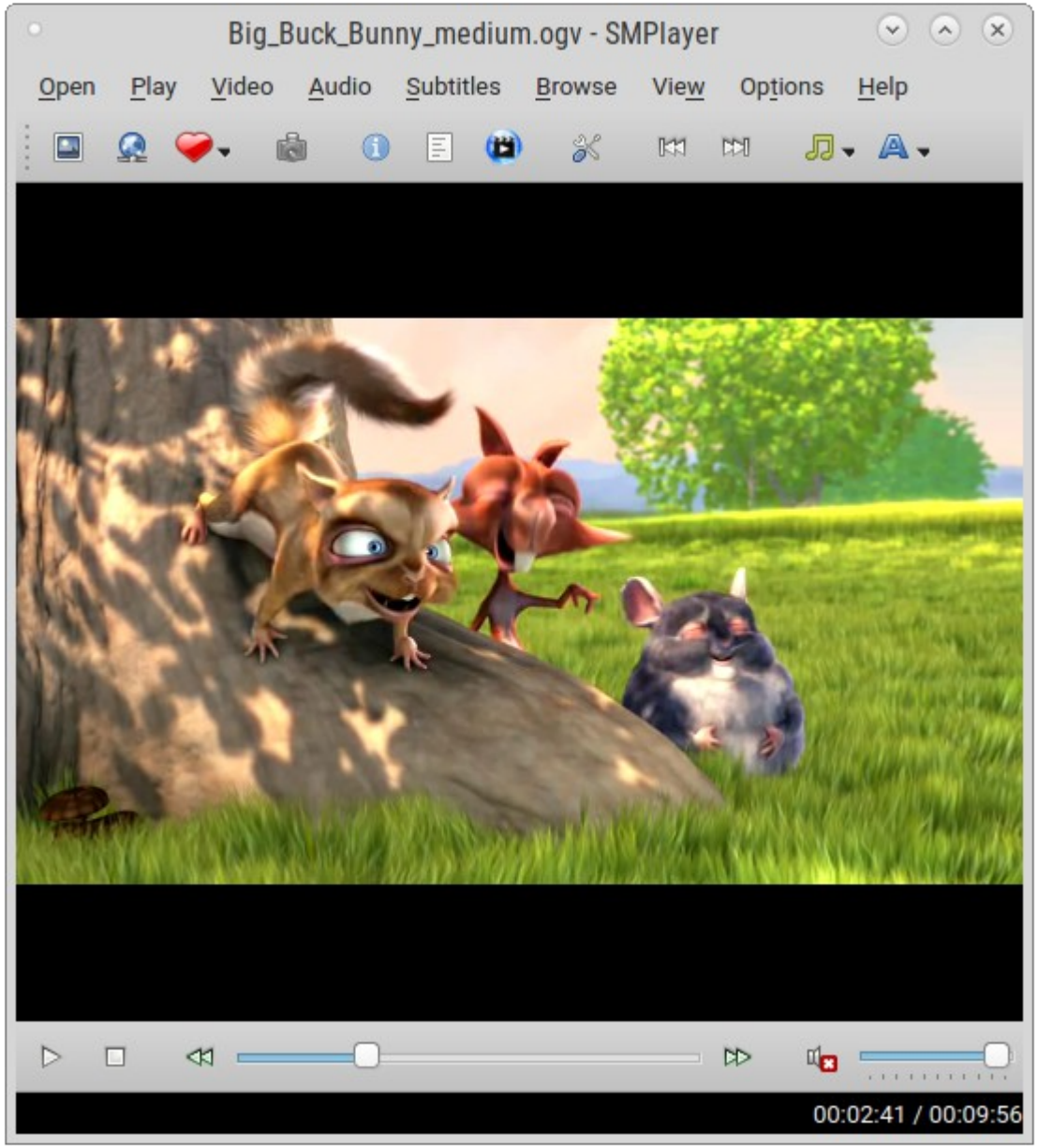
```
char data = ...
size_t size = ...

write(fd, data, size);
```

Обмен данными активно задействует ядро

```
ssize_t sendfile(int out_fd, int in_fd,
                 off_t *offset, size_t count)
```

- Нестандартный системный вызов Linux
- Исключает многократное копирование данных между пространством пользователя и пространством ядра

- Не очень удобен, если данные генерируются on-the-fly



Задействованные процессы:

1. X-сервер
2. консольный плейер mplayer
3. GUI для mplayer

Взаимодействие:

1. Между mplayer и GUI - через каналы
2. Между mplayer и X - *разделяемая память*

Разделение памяти

Физический адрес (20 бит)	Резерв (3 бита)	Флаги (9 бит)

G	0	D	A	C	W	U	R	P

- G (global) - страница глобальная
- D (dirty) - страница была модицифирована
- A (accessed) - был доступ к странице
- C (caching) - запрещено кеширование
- W (write-throught) - разрешена сквозная запись
- U (userspace) - есть доступ обычному процессу
- R (read+write) - есть права на запись
- P (present) - страница находится в физической памяти

Разделение памяти

- Модуль MMU (Memory Management Unit) выполняет вычисления адресов по таблицам
- Кеш TLB (Translation Lookaside Buffer) хранит таблицы памяти для текущего процесса

Поддержка на уровне ОС

Unix System-V

SVR4 (1988) - Inter Process Communications:

- Shared Mem
- Semaphores
- Message Queues

BSD UNIX

4.3BSD (1986) - реализация mmap

Разделяемая память

System-V style

- **`ftok`** - связать "имя файла" с некоторым ключем `key_t`
- **`int shmget(key_t,sizet_t,int flags)`** создать или открыть сегмент разделяемой памяти
- **`void* shmat(int id, void *addr, int flags)`** присоединить сегмент в адресное пространство текущего процесса

Разделяемая память

System-V style

- Используется отдельное пространство имен-ключей
- Ядро хранит отдельную таблицу сегментов

- Память доступна всем процессам по имени
- **Последний процесс должен удалить разделяемый сегмент**

Разделяемая память
BSD style

```
void *mmap(void *addr, size_t length,
           int prot, int flags,
           int fd, off_t offset);
```

- `addr` - рекомендуемый адрес размещения
- `length` - размер, кратный размеру страниц, который можно получить как `sysconf(_SC_PAGE_SIZE)`
- `prot` - флаги защиты памяти `(EXEC/READ/WRITE/NONE)`
- `flags` - флаги доступа
  - **`MAP_SHARED` – совместный доступ разным процессам,**
  - **`MAP_PRIVATE` – использование Copy-On-Write**
- `fd, off_t` - файл для отображения (если отображение на файл)

Разделяемая память
BSD style

- Параметр MAP_ANONYMOUS
- Позволяет указать -1 в качестве файлового дескриптора отображаемого файла → данные не будут сохраняться в файл

Разделяемая память
BSD style, именование файлов

- Создаем файл, доступный нескольким процессам
- Открываем его через mmap
- Не злоупотребляем msync
- PROFIT!

- кто-то должен потом файлик удалить
- нужно место на диске

Удаление файла

- Файл - это пара <dev_id, inode>
- У каждого файла есть аттрибут nlink
- Операция "удаления" - это nlink--
- Файл становится не доступен при:
  - nlink==0
    &&
  - закрыты все файловые дескрипторы во всех процессах

Файловая система tmpfs

- Временное хранилище, все файлы будут удалены при выключении
- Имеет фиксированный размер, который легко изменить без потери данных
- Занимает в памяти столько, сколько данных записано
- Может задействовать swap, если память закончилась

```
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
```

POSIX shm

- `int shm_open(name,flags,mode)` аналогично open внутри /dev/shm
- Ограничения на имя:
  - начинается с "/"
  - максимальная длина - NAME_MAX
- Режимы открытия:
  - O_RDONLY
  - O_RDWR
- Функция glibc, а не системный вызов

Гонка данных

Проблема гонки данных

1 процессор

- Переключение планировщиком задач активного потока выполнения

N процессоров

- Одновременное выполнение нескольких потоков выполнения

Гонка данных и shared mem

- Несколько процессов имеют одну область памяти в своем адресном пространстве
- Никто не знает, что делают другие
- Требуется побочный канал связи для синхронизации
- **Сигнал? Канал?**

Семафор

- Целочисленный счетчик (беззнаковый)
- Операции:
  - увеличить значение (освободить)
  - попытаться уменьшить значение (захватить); если значение счетчика уже равно 0, то ждать, пока кто-то другой его не освободит
- Операции инкремента/декремента - **атомарные**

Атомарность объекта

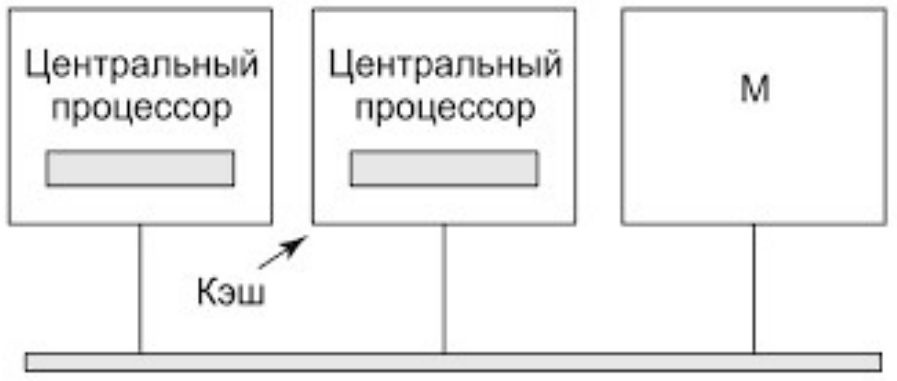
Необходимые условия:

1. Умещается в размер машинного слова
2. Выровнен по границе машинного слова

Пример: `sig_atomic_t`

Разделяемая память в UMA-системах [**U**niform **M**emory **A**ccess]

- Все процессоры (ядра) используют общую шину памяти
- При одновременном доступе шина памяти блокируется → простой всех ядер
- Проблема решается наличием отдельного кэша для каждого ядра



Уровни кэша (в x86)

- L1 (index0 + index1 для Intel) - самый близкий кэш микроинструкций и текущих данных, с которыми оперируют микроинструкции
- L2 - общий кэш, связанный с ограниченным количеством ядер (как правило - одним, но не всегда)
- L3 - общий кэш, связанный со всеми ядрами

Информация в Linux

- `/sys/devices/system/cpu/cpu0/cache`
  - `index0`
  - `index1`
  - `index2`
  - `ibdex3`
- cpuX - отдельное ядро (они одинаковые, так что всё равно)
- indexY - отдельные кэши процессора

Характеристики кэшей

- Уровень в иерархии кэшей [`level`]: число от 0 до 2
- Размер кэша [`size`]: от 64Кбайт (L1) до 35Мб (L3 в топовых Intel Xeon, их тех, что сейчас можно купить в Москве, для Core-серии характерный размер 8Мб)
- Тип [`type`]: Data,Instruction или Unified
- Какими ядрами используется [`shared_cpu_list`]: номера ядер

Как устроен кэш

- **Блок** - минимально адресуемый объем данных в кэше. 64 байта для Intel
- **Кэш-линия** - блок + метаданные, определяющие адрес в памяти
- **Набор** - связан с некоторым адресом в оперативной памяти. Для L3 - 12 линий по 64 байт = 768 байт
- Весь кэш состоит из независимых наборов. Для Core i3 размер кэша L3 3Мб = 64 байта в блоке * 12 линий в наборе * 4096 наборов

Атомарность объекта

Необходимые условия:

1. Умещается в размер машинного слова
2. Выровнен по границе машинного слова
3. **Должен быть одинаковым для всех ядер в любой момент времени**

Для x86: есть флаг, запрещающий кэширование страницы

Семафоры POSIX

```
#include <semaphore.h>
```

- Атомарные счетчики `sem_t*`, которые хранятся в специальных страницах памяти
- Реализованы операции:
  - `sem_post` - освободить семафор (+1)
  - `sem_wait` - захватить семафор (-1)
  - `sem_getvalue` - прочитать значение, не блокировать

Семафоры POSIX

- Безымянные (для использования несколькими нитями внутри процесса или родственными процессами)

```
sem_init(sem_t *sem,
         /* 0 - в MAP_PRIVATE,
            1 - в MAP_SHARED */
         int pshared,
         uint32_t initial)
```

Семафоры POSIX

```
sem_t* sem_open
```

- аналогично shm_open - открывает или создает семафор в /dev/shm/