Московский Физико-Технический Институт
Физтех-школа Аэрокосмических технологий
Институт Аэромеханики и Летательной Техники

МФТИ

ФАКТ

# Архитектура Компьютера и Операционные Системы. Часть 2. Основы операционных систем

# Лекция 5: Средства IPC. Разделяемая память

Новиков Андрей Валерьевич
д.ф.-м.н.

Жуковский

# Недостатки потокового обмена данными между процессами

- Потоковая передача между процессами – pipe, FIFO
- Системные операции чтения и записи не анализируют передаваемые данные.
  - неизвестно записаны данные одним процессом или несколькими;
  - неизвестно данные записаны за один раз или в нескольких операциях.
- Требуется как минимум 2 операции копирования данных:
  - из передающего процесса в системный буфер (память ядра);
  - из системного буфера в принимающий процесс.
- Процессы, обменивающиеся данными, должны существовать одновременно.

# System V IPC

- Эффективные механизмы передачи данных между процессами (**i**nter**p**rocess **c**ommunications)
- Начали появляться в Unix System V (1983). Перешли во все современные ОС под названием SystemV IPC.
- Средства связи System V IPC:
  - разделяемая (общая) память (shared memory);
  - семафоры (semaphores);
  - очереди сообщений (message queue).
- Общность происхождения, схожий интерфейс.

# Пространство имён

- Средства связи из группы System V IPC основаны на непрямой адресации.
- Средство связи должно иметь имя.
- Множество всех возможных имён объектов – пространство имён для данных объектов.
  - для FIFO пространство имён — множество допустимых имён файлов файловой системы;
  - для объектов SystemV IPC пространство имён — множество значений целочисленного «ключа» **key_t**.

# Пространство имён System V IPC

- Каждый объект SysV IPC имеет уникальное имя – ключ **key_t** (целочисленный тип данных)

- Присваивать непосредственно нельзя, генерируется на основе комбинации:
  - имя существующего файла, доступного для чтения и не меняющего расположения на диске;
  - произвольное целое число (локальный номер средства связи).

- Обеспечивается уникальность ключа, т.к. полное имя файла уникально. На основе одного файла можно сделать несколько ключей.

# Генерация ключа SysV IPC

```
#include <sys/types.h>
#include <sys/ipc.h>

key_t ftok(char* path, char proj);
```

- Возвращаемое значение
  >0 – ключ, обычно целое 32-бит
  -1 – ошибка
- **path** – имя некоторого файла, доступного для чтения. *Содержимое файла никак не используется, важно только имя!*
- **proj** – целое число, характеризующее экземпляр средства связи

# Дескрипторы System V IPC

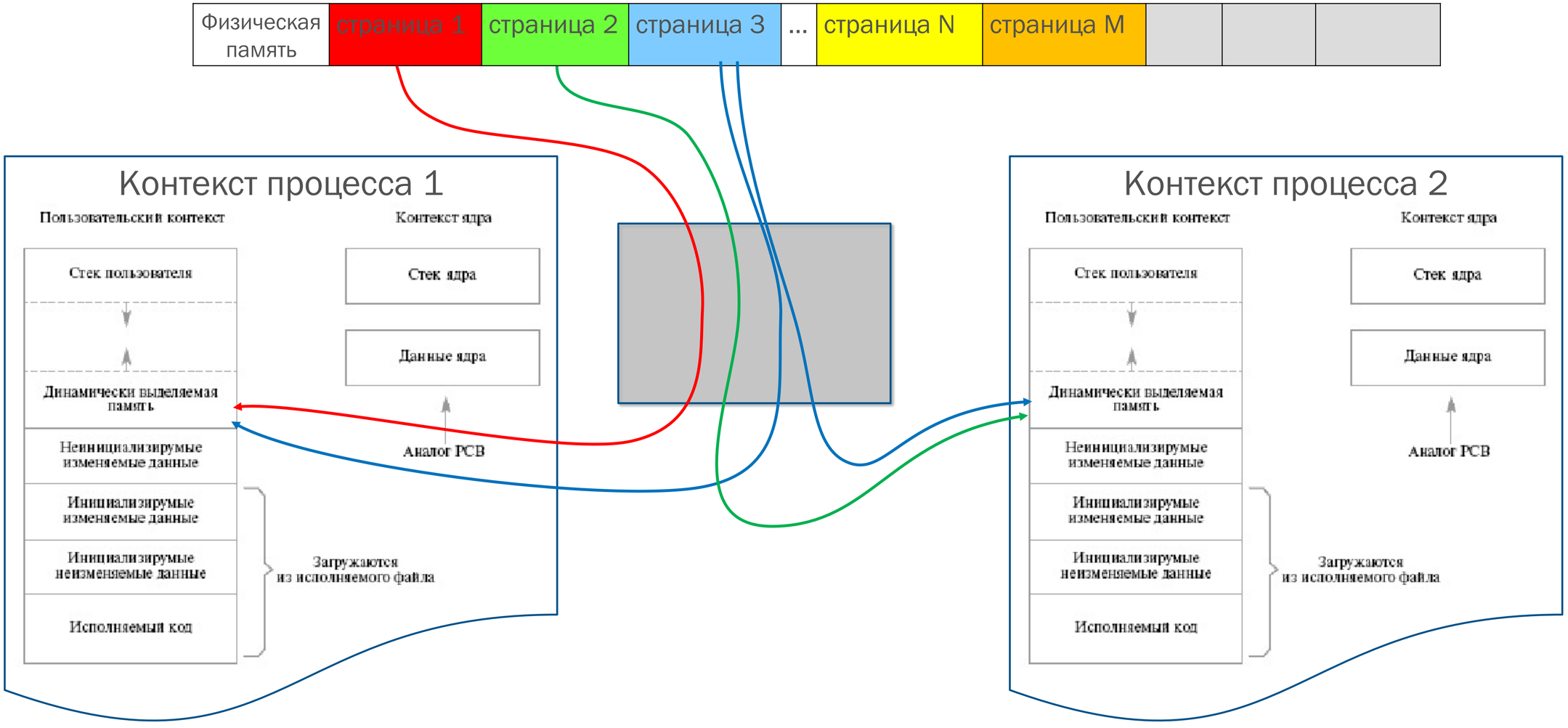
## ❑ Дескриптор SysV IPC

- используется для идентификации средства связи во всех операциях обмена данными
- Уникальный во всей системе
- Относится к глобальной таблице объектов SysV IPC, хранящейся вне контекста процессов.
- Не закрывается после завершения процесса.

## ❑ Файловый дескриптор

- используется для идентификации потока (средства связи) во всех операциях чтения/записи
- локальный для процесса
- относится к локальной таблице открытых файлов, хранящейся в системном контексте процесса
- закрывается при завершении процесса

# Страничная организация памяти



# Совместно используемая память (shared memory)

- Позволяет разным процессам совместно использовать область оперативной памяти.

- Типичная последовательность действий для работы с shared memory.
  - 0. **key = ftok(...)** - сгенерировать уникальное имя (ключ)
  - 1. **shmid = shmget(key,...)** – получить доступ к объекту (открыть), получить дескриптор SysV IPC
  - 2. **mem = shmat(shmid, ...)** – подключить область памяти в контекст процесса, как будто выделить помощью malloc(), new и т.п.
  - 3. Использовать память как обычно.
  - 4. **shmdt(mem)** – отключить память, как при free() или delete.

# Открытие (создание) совм.используемой памяти

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

int shmget(key_t key, int size,
           int shmflg);
```

- Возвращаемое значение
  >0 – дескриптор SysV IPC
  -1 – ошибка
- **key** – уникальное имя (ключ), созданное **ftok()** или *IPC_PRIVATE*
- **size** – размер в байтах создаваемого или существующего сегмента памяти
- **shmflg** – флаги, комбинируются с помощью **|** «битовое или»

# Открытие (создание) совм.используемой памяти

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>


int shmget(key_t key, int size,
           int shmflg);
```

- shmflg – флаги, комбинируются с помощью | «битовое или»
  - IPC_CREAT - создать если не существует
  - IPC_EXCL - вместе с предыдущим, создавать эксклюзивно, ошибка если сущуствует
  - Права доступа
  - 0400 – только чтение для владельца
  - 0200 – только запись для владельца
  - 0100 – только исполнение для владельца
  - 0040 – только чтение для группы
  - …

# Создание разделяемой памяти. Пример

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>


int main(int argc, char* argv[], char* envp[])
{
    key_t k = ftok(“mylabel”, 0);  if( k < 0 ){ … }
    int shmid = shmget(k, 3*sizeof(int), IPC_CREAT|0664);  if( shmid < 0 ){ … }
    //…
    return 0;
}
```

# Подключение сегмента памяти

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

void* shmat(int shmid, char
*shmaddr, int shmflg);
```

- Возвращаемое значение
  >0 – указатель в адресном пространстве процесса
  (void*)(-1) – ошибка
- **shmid** – дескриптор SysV IPC
- **shmaddr** – желаемый адрес в памяти, либо NULL, если автоматически
- **shmflg** – флаги
  - 0 –чтение и запись
  - SHM_RDONLY – только чтение

# Отключение сегмента памяти

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

int shmdt(char *shmaddr);
```

- Возвращаемое значение
  =0 – успех
  -1 – ошибка
- shmaddr –адрес в памяти, по которому подключен сегмент

# Удаление совм.используемой памяти

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>


int shmctl(int shmid, int cmd,
struct shmid_ds *buf);
```

Управление сегментом памяти

- Возвращаемое значение
  =0 – успех
  -1 – ошибка
- **shmid** – дескриптор памяти SysV IPC
- **cmd** – код команды
  **IPC_RMID** –сегмент памяти помечается на удаление, будет удалён после отключения везде
- **buf** – данные для команды управления
  - **NULL** для удаления

# POSIX Shared memory

- Более современный способ работы с совместно используемой памятью.
  Альтернатива SystemV Shared memory.
  В рамках нашего курса НЕ ИСПОЛЬЗУЕМ, но имеем ввиду.

- Пространство имён объектов памяти = имена файлов из файловой системы
- **shm_open()**: создать или открыть объект разделяемой памяти по имени файла (спец. файл обычно в /dev/shm/ на файловой системе tmpfs).
- **mmap()**, **munmap()**: подключить/отключить объект разделяемой памяти в адресное пространство (как при отображении обычного файла в память).
- **shm_unlink()**: удалить объект разделяемой памяти (спец. файл).

# Консольные команды SysV IPC

```
$ ipcs –t|-p|-c|-l|-u …
```

Информация об объектах SysV IPC

- **-t** – показать время последней операции
- **-p** – PID процесса создателя и совершившего последнюю операцию
- **-l** – системные лимиты для объектов SysV IPC
- **-u** – суммарное состояние (summary)

# Консольные команды SysV IPC

```
$ ipcrm –m|-M|-q|-Q|-s|-S
```

Удаление объектов SysV IPC

- -m id – удалить память по идентификатору
- -M key – удалить память по ключу
- -q / -Q – удалить очередь сообщений по идентификатору/ключу
- -s / -S – удалить группы семафоров по идентификатору / ключу