

Прикладные физико-технические и компьютерные методы исследований

Семинар 5

https://dl.dropboxusercontent.com/u/96739039/infa_s05.pdf

Средства коммуникации процессов

- Сигнальные
- Канальные (***потокковая модель***
(pipe+работа с файлами), сообщения)
- Разделяемая память

FIFO – именованный pipe

- Pipe позволяет общаться **только родственным** процессам благодаря тому, что таблица файловых дескрипторов (часть системного контекста) наследуется дочерним процессом.
- Аналог pipe для общения неродственных процессов – FIFO
- Данные о расположении и состоянии канала связи FIFO можно узнать через файловую систему (записаны в файле на диске)
- Пример: osstud/05/stud/05-4c.htm

Особенности open для FIFO

- Без дополнительных флагов ... при открытии FIFO на запись вызов open блокируется, пока кто-то ещё не откроет её на чтение
- И наоборот, при открытии FIFO на чтение вызов open блокируется, пока кто-то не откроет её на запись

Упражнение 1

- Создать текстовый мессенджер, позволяющий двум неродственным процессам отправлять сообщения друг другу через FIFO
- Для считывания строки из терминала используйте `fgets`. 3м параметром передавайте `stdin`

Комментарии к упражнению 1

- **fgets** для считывания текста из терминала блокирующий + **read** для считывания из fifo – тоже блокирующий (как и для pipe'a) ... и как с этим работать? ...
- не забывайте про особенность open для fifo
- выводите на экран всю отладочную информацию: имена fifo, которые пытаемся создать/открыть, номера соответствующих дескрипторов и т.д.
- код для клиента №0 и клиента №1 очень похож -> можно обойтись одним файлом с кодом ... но запуск клиентов производить так:

```
./a.out 0
```

и

```
./a.out 1
```

- mknod вернёт ошибку, если файлы для работы с fifo уже были созданы ранее. Проверить данный случай можно с помощью кода

```
if (errno != EEXIST)
```

```
{
```

```
....
```

```
}
```

Указатели на функции

```
typedef int (*MyFunc)(int, char);  
int g(int x, char c) {  
    return x + c;  
}  
void f(MyFunc func) {  
    int result;  
    result = func(42, 'z');  
    ...  
}  
int main() {  
    f(g);  
}
```

Позволяют лишний раз не дублировать код программы.

Упражнение 2

- Реализовать ф-ю сортировки массива, принимающую на вход ф-ю сравнения: с помощью одной ф-и сортировки и двух разных ф-й сравнения отсортировать массив по возрастанию/убыванию.

Нити исполнения (threads)(легковесные процессы)

- Разделяют
 - Программный код (пользовательский)
 - Глобальные переменные (пользовательский)
 - Все нити живут в одном адресном пространстве – «куча» общая (пользовательский)
 - Системные ресурсы (таблица открытых файлов, текущая директория и т.п.)
 - Но каждая нить имеет собственный
 - Программный счётчик
 - Содержимое регистров
 - Стек
- т.е. нити могут работать параллельно, и контекст нитей переключается быстрее.

Нити исполнения (threads)

- Общие глобальные переменные – можно использовать для коммуникации между нитями
- В Linux posix thread`ы реализованы не эффективно, поэтому выигрыш в производительности можно и не заметить

Нити исполнения (threads)

- ***void* thread(void* arg);***
- **pthread_create** (fork)
- **pthread_exit** (exit)
- **pthread_join** (waitpid)
- В случае ошибки возвращается число > 0
(ранее всегда < 0 обозначало ошибку)

См. пример

<https://dl.dropboxusercontent.com/u/96739039/s07e01.c>

gcc main.c -lpthread

Нити исполнения (threads)

- **int pthread_create(pthread_t *thread, pthread_attr_t *attr, void * (*start_routine)(void *), void *arg);**
- **void pthread_exit(void *status);**
- **int pthread_join (pthread_t thread, void **status_addr);**

Упражнение 3

- Модифицируйте предыдущий пример, добавив в него 3-ю нить исполнения

Упражнение 4 (Состояние гонки)

- Создаём массив из 3-х переменных. Создаём две нити.
- 1-я в цикле 10^7 раз увеличивает на единицу $a[0]$ и $a[2]$
- 2-я в цикле 10^7 раз увеличивает на единицу $a[0]$ и $a[2]$
- Основная нить дожидается завершения обеих
- Выводим $a[0]$, $a[1]$, $a[0] + a[1]$, $a[2]$

Упражнение 4 (домашнее: одно из 2х) (необходимо сдать через 2 недели)

1. Найти произведение двух больших квадратных матриц с помощью N потоков. Размеры матрицы кратны N .
2. Есть большое количество экспериментальных данных (нужно самим сгенерировать)
 $\sim 10^7 - 10^8$ вещественных чисел. Необходимо с помощью N потоков найти их среднее и дисперсию.

С помощью утилиты `time` измерить время работы программы в зависимости от N , либо ϕ -и `time`.