Прикладные физико-технические и компьютерные методы исследований

Семинар 6

https://dl.dropboxusercontent.com/u/96739039/infa_s06.pdf

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 (*).

Средства коммуникации процессов

- Сигнальные
- Канальные (***потоковая модель*** (работа с файлами/pipe/fifo), сообщения)
- Разделяемая память

Минусы потоковой коммуникации

- Нельзя понять, данные передал один или несколько процессов, за один раз или в несколько подходов и т.п.
- Требуется минимум 2 операции копирования (адр. пр.1 -> адр. пр. ОС -> адр. пр. 2)
- Процессы должны существовать одновременно.

System V IPC (InterProcess Communication)

- Разделяемая память
- Семафоры
- Очереди сообщений

Адресация в System V IPC

- непрямая адресация (как и у pipe/fifo) – явно не указываем участников общения
- для общения неродственных процессов у средства коммуникации должно быть «имя». (у fifo – имя файла в файловой системе).
- пространство имён

Пространство имён

key_t ftok(char *pathName, char proj);

К pathName имеется доступ на чтение.

Некоторые ОС учитывают также содержание файла, а не только полный путь, поэтому **НЕ нужно** использовать исполняемый файл для генерации ключа

Участок разделяемой памяти расположен в адресном пр-ве ОС, а не в этом файле!

Дескрипторы System V IPC

- Для файлов, pipe и fifo информация о них хранится в таблице файловый дескрипторов, которая уничтожается при завершении процесса.
- Информация о System V IPC хранится в адресном пространстве ОС, поэтому может «жить» дольше процесса

Разделяемая память (shared memory)

- ftok -> имя средства коммуникации
- shmget -> дескриптор существующего, либо созданного средства
- shmat -> прикрепление адресного пр-ва ОС к адресному пр-ву процесса
- shmdt -> откреплениe …

Пример подсчёта количества запусков программ

https://dl.dropboxusercontent.com/u/96739039/s06e01a.c

https://dl.dropboxusercontent.com/u/96739039/s06e01b.c

Удаление разделяемой памяти

- ipcs -m

информация обо всех ресурсах System V IPC

- ipcrm shm <id>

удаление разделяемой памяти

- либо программно

shmctl(sharedMemoryID, IPC_RMID, NULL);

fork, exec, exit

- fork – все участки разделяемой памяти наследуются
- exec/exit – все участки разделяемой памяти исключаются из адресного пр-ва процесса, но продолжают существовать в системе

Упражнение 1

Напишите две программы, осуществляющие взаимодействие через разделяемую память. Первая программа должна создавать сегмент разделяемой памяти и копировать туда свой собственный исходный текст, вторая программа должна брать оттуда этот текст, печатать его на экране и удалять сегмент разделяемой памяти из системы.

Семафоры

Необходимость синхронизации процессов + условие гонки

- A(S, n) – увеличить значение семафора на n
- D(S, n) - пока значение семафора S < n

процесс блокируется. Далее S = S - n

- Z(S) - процесс блокируется до тех пор, пока значение семафора S не станет равным 0

- IPC семафоры при создании инициируются нулевым значением

Создание массива семафоров

- **int semget(key_t key, int size, int shmflg);**

Работа с семафорами

- **int semop(int semid, struct sembuf *sops, int nsops);**

sembuf – описывает действия для каждого семафора (там указывается номер sem_num, тип операции sem_op: -n, 0, n и флаги sem_flg (будем всегда задавать 0).

Пример программы с использованием семафоров

https://dl.dropboxusercontent.com/u/96739039/s08e01a.c

https://dl.dropboxusercontent.com/u/96739039/s08e01b.c

Первая D(S, 1)

Вторая A(S, 1)

Упражнение 2

Внести правки в пример, чтобы программа могла работать без блокировки после не менее 5 запусков второй программы.

Удаление семафоров

- ipcs -s

информация обо всех ресурсах System V IPC

- ipcrm sem <id>

удаление семафора

- либо программно

semctl(semID, IPC_RMID, NULL);

Упражнение 3

- Два процесса (неродственных) «логгируют» некую информацию в один файл (каждая пишет свою строчку в файл)
- С помощью семафора обеспечить отсутствие условия гонки.

Прикладные физико-технические и компьютерные методы исследований

Семинар 6 (продолжение)

https://dl.dropboxusercontent.com/u/96739039/infa_s06.pdf

Упражнение 4

- Организуйте двунаправленную связь через один pipe, определяя очередность работы с помощью семафора: хотим создать **конвейер, используя для синхронизации** два **семафора.**

```
while (1)                          while (1)
{                                  {
        write a, b                                 read a, b
        read answer                                write answer = a + b
}                                  }
```

Упражнение 4

S1 = 0, S2 = 1

```
while (1)                          while (1)
{                                  {
        write a, b                         D(S2, 1)
        D(S1, 1)                           read a, b
        read answer                        write answer = a + b
        A(S2, 1)                           A(S1, 1)
}                                  }
```

Dead lock

Реализуйте и проанализируйте программу, работающую с двумя семафорами: каждый цикл работает в своём потоке: изначально

S1 = S2 = 1

```
while (1)
{
        D(S1, 1);
        D(S2, 1);
        A(S1, 1);
        A(S2, 1);
}
```

```
while (1)
{
        D(S2, 1);
        D(S1, 1);
        A(S1, 1);
        A(S2, 1);
}
```

Dead lock

Пусть работа семафоров логгируется в файл: записываются события

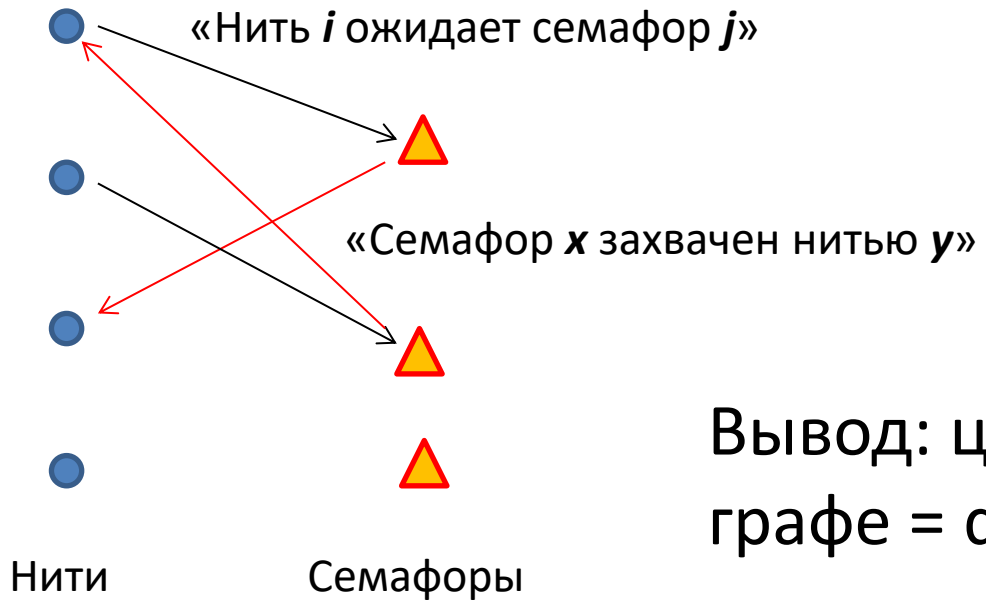
1) поток ***i*** ждёт семафор ***j*** (операция D)
2) семафор ***j*** захвачен ***i-м*** потоком.

Как по такому логу понять, надо ли «убивать» программу, т.к. висим в deadlock’е или стоит подождать из-за «медленн(ой)ых» нитей?

P.S. Вместо семафоров могут быть mutex’ы – суть та же.

Dead lock

- Нужно составить граф … двудольный



Вывод: цикл в таком графе = deadlock

Проверить граф на цикличность можно с помощью обхода в глубину (DFS): http://e-maxx.ru/algo/finding_cycle … DFS – очень полезный и часто применяемый алгоритм

P.S. Можете сами дома реализовать такое логгирование и анализ лога … по желанию

Домашнее упражнение (на 2 недели)

- Написать упражнение «про посуду», общение между процессами осуществляется одним (любым) известным вам способом (shared memory, pipe/fifo, обычный файл на диске). Используйте семафоры для синхронизации процессов.