Лабораторная работа №1
Средства межпроцессного взаимодействия ОС UNIX/Linux: семафоры и разделяемая память. Модель “производитель-потребители”.

Теоретическая часть

Понятие процесса является основополагающим для операционных систем семейства UNIX.

Процесс - это совокупность набора инструкций программы, запущенной на выполнение, а также среды или окружения, в которой осуществляется выполнение программы.

Среду выполнения образуют ресурсы памяти, доступные системные ресурсы и устройства ввода/вывода. Процесс также можно определить, как совокупность данных ядра системы, необходимых для описания образов программ в памяти и управления их выполнением. Структуры данных, описывающих различные процессы, а также образы программ в памяти, соответствующих разным процессам, никогда не пересекаются.

Процесс может считывать и записывать информацию в выделенные ему раздел данных и стек, но ему не доступны данные и стеки других процессов.

Образ текущего состояния процесса называется его контекстом. Наличие контекста позволяет реализовать переключение выполнения с одного процесса на другой, и тем самым обеспечивать режим многозадачности даже для однопроцессорных ЭВМ.

Для идентификации процессов каждому из них присваивается уникальный идентификатор, или номер: pid. С каждым из процессов связывается набор атрибутов, позволяющих провести дополнительную идентификацию или определить свойства процесса. Такими атрибутами являются: идентификатор процесса-родителя (ppid), идентификатор сеанса, идентификатор ассоциированного псевдотерминала (tty), реальные и эффективные идентификаторы пользователей и групп.(ruid, rgid, euid, egid), приоритет процесса (nice number) и т.д.

Обеспечение взаимодействия процессов осуществляет модуль, входящий в состав подсистемы управления процессами. В UNIX-системах поддерживаются следующие средства межпроцессных коммуникаций, или IPC (от Inter-Process Communications).:

-сигналы;

-неименованные каналы;

-именованные каналы, или FIFO (происходят из системы System III);

-файлы;

-группа средств, пришедших из архитектуры UNIX System V (семафоры, разделяемая память, очереди сообщений);

-средства, унаследованные от UNIX BSD (сокеты).

Сигналы, неименованные каналы, FIFO и обычные файлы относят к простейшим средствам межпроцессного взаимодействия.

Сигналы - это сообщения, генерируемые операционной системой в ответ на происходящие события.

Неименованные каналы реализуются в виде пары файловых дескрипторов, один из которых используется для чтения информации из канала, другой - для записи в канал. Неименованные каналы позволяют обмениваться информацией только родственным процессам, поэтому определение неименованного канала как правило предшествует вызову функции порождения процессов fork. В классическом варианте передача информации через неименованный канал возможна только в одном направлении.

Именованные каналы реализуются как файлы особого типа: FIFO. Эти средства позволяют организовать обмен информацией между любыми, даже и не родственными процессами, выполняющимися в одной файловой системе. Отличительным свойством FIFO является нулевая длина файлов этого типа, поэтому в них нельзя хранить информацию.

Обычные файлы являются наиболее очевидным средством для обмена информацией между процессами. Однако с использованием файлов одновременно несколькими процессами связана проблема гонок. Для разрешения этой проблемы можно использовать функцию flock, которая задает дисциплину доступа к открытому файлу, однако данное средство следует отнести к ненадежным.

В UNIX System V впервые появились такие средства межпроцессного взаимодействия, как семафоры, очереди сообщений и разделяемая память. Созданию этих объектов предшествует генерация уникального системного идентификатора (ключа), который регистрируется IPC и позволяет обращаться к данным объектам из различных (в том числе и не родственных) процессов. В целом, функции для работы с данными средствами межпроцессных взаимодействий, подразделяются на три группы:

-функция генерации ключа ftok;

-управляющие функции семейств *get и *ctl (создание и уничтожение объекта, задание и изменение его базовых характеристик);

-все прочие функции, изменяющие состояния объекта (например, добавление сообщения в очередь, освобождение или захват семафора, и т.д.).

Сообщения помещаются в очередь сообщений и являются разделяемыми системными ресурсами. Процесс может работать с несколькими очередями, каждая из которых имеет свой собственный идентификатор. Сообщения реализуются как структуры, включающие в себя: тип сообщения; длину сообщения; собственно данные. Сообщения можно мультиплексировать, т.е. в одну и ту же очередь помещать сообщения для разных процессов.

Разделяемая память - это область оперативной памяти произвольной структуры, доступная для чтения-записи одновременно нескольким

процессам по уникальному идентификатору (ключу), как правило, используется совместно с семафорами.

Семафор в UNIX реализуется как группа из нескольких счетчиков, объединенных общими признаками, например дескриптором объекта, правами доступа, и т.п. Каждый из этих счётчиков может принимать любое неотрицательное значение в пределах определённой системы. Операционная система не накладывает семантических ограничений на использование различных состояний семафоров. В частности, процессы могут решать, какое значение семафора считать разрешающим, на какую величину изменять значение семафора и так далее.

Одним из простейших вариантов использования семафоров UNIX является бинарный семафор Дейкстра, но возможны и более сложные схемы.

Сокеты – средства межпроцессорного взаимодействия, появившиеся в BSD UNIX. Назначение сокетов состоит в обеспечении унифицированного интерфейса обмена информацией между процессами, вне зависимости от того, выполняются ли они в одной файловой системе или в разной.

Сокеты, в зависимости от основных характеристик, могут принадлежать к одному из двух доменов: UNIX или Internet. Сокеты UNIX реализуются как файлы особого типа и являются дальнейшим развитием концепции FIFO. Сокеты Internet описывают сетевое соединение и идентифицируются парой, состоящей из IP-адреса и номера порта

В данной работе требуется реализовать взаимодействие между параллельными процессами по модели “производитель-потребители”. В этом случае имеется процесс-производитель, который формирует данные, помещаемые в разделяемую область памяти (общий ресурс). Остальные процессы выполняют роль потребителей, захватывая общий ресурс и извлекая данные из разделяемой области памяти, при этом извлеченные данные уничтожаются.

Очевидно, что работа с разделяемым ресурсом должна быть организована в соответствии с моделью семафора Дейкстра, который позволяет управлять доступом к разделяемому ресурсу, избегая ситуаций блокировок и отталкивания.

Механизм семафора Дейкстра включает в себя переменные особого типа - семафоры, и две операции - P и V, единственным аргументом которых может быть только переменная типа семафор. Состояние семафора представляется целым неотрицательным числом. Если диапазон значений ограничен нулем и единицей, то семафор называют бинарным.

В соответствии с механизмом семафора Дейкстра, операция P выполняется процессом перед входом в критическую секцию, а операция V - перед выходом из критической секции.

Операция P реализуется по следующему алгоритму:

- если значение семафора отлично от нуля, оно уменьшается на единицу;

- если значение семафора равно нулю, то операция P не завершается до тех пор, пока другой процесс не увеличит значение семафора с помощью

операции V, после чего P продолжит выполнение и уменьшит значение семафора на единицу.

Операция V всегда увеличивает значение семафора на единицу.

Операции P и V являются неделимыми в том смысле, что их выполенение не может быть прервано на какой-либо промежуточной фазе. Тем самым гарантируется целостность состояния переменной-семафора и отсутствие конфликтных ситуаций.

На рисунке 1 показан пример взаимодействия двух процессов, организованного с помощью семафора Дейкстра. Переходы, помеченные p_1, p_2 соответствуют выполнению операций P, переходы, помеченные v_1, v_2 - выполнению операций V. Выполнению операций критической секции соответствуют переходы c_1, c_2, а остатки циклов моделируются переходами r_1, r_2.

Экспериментальная часть.

1. Разработать программу на языке C/C++ для ОС UNIX/Linux, которая моделирует обслуживание клиентов бензозаправочной станции.

На бензозаправочной станции имеется пять колонок с бензином АИ-76 (2 колонки), АИ-92 (2 колонки) и АИ-95 (1 колонка). Прибывающие на заправку машины ожидают обслуживания в общей очереди, которая программно моделируется областью разделяемой памяти. При поступлении новой заявки на обслуживание (новой машины) в область разделяемой памяти добавляется запись, содержащая:

-уникальный порядковый номер машины (клиента);

-требуемая марка бензина (одна из перечисленных).

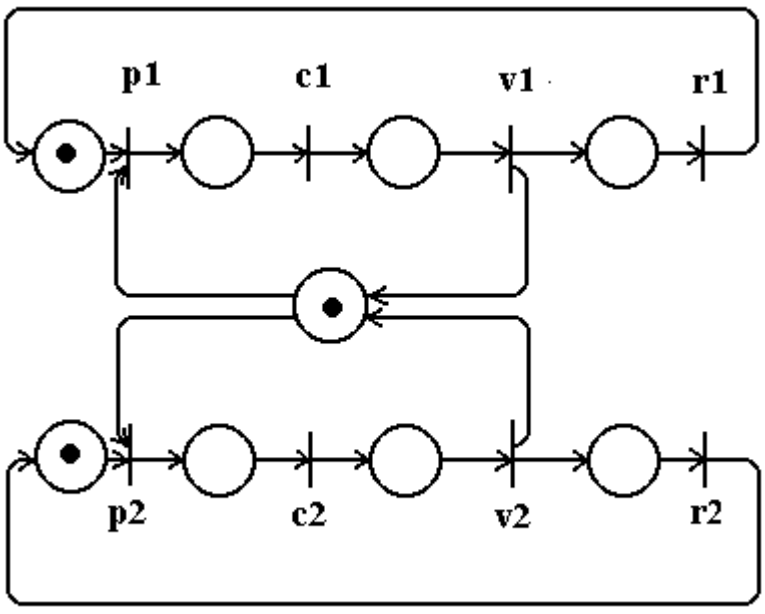

Рисунок 1 - Синхронизация двух процессов с помощью семафора Дейкстра.

Состояние области разделяемой памяти отображается в протоколе, который может быть реализован как текстовый файл. Протокол содержит данные о всех поступавших заявках в формате, аналогичном записям, помещаемым в область разделяемой памяти.

Как только какая-либо бензоколонка освобождается, очередь покидает машина, для которой требуется бензин соответствующей марки и которая ближе остальных находится к началу очереди. Эта машина занимает соответствующую бензоколонку. При этом информация о клиенте удаляется из области разделяемой памяти, моделирующей общую очередь. После обработки заявки информация о клиенте переносится в файл протокола (текстовый файл), хранящий информацию о клиентах, обслуженных на данной бензоколонке.

Таким образом, в приложении используется шесть файлов протокола (соответствуют очереди и бензоколонкам) и шесть потоков (см. рисунок 2):

-поток заявок, поступающих в очередь;

-потоки, моделирующие обслуживание заявок.

2. Формат записи информации выбирается разработчиком. Информация из файла протокола не удаляется, обслуженные заявки каким-либо образом помечаются.

Эксперимент проводить для не менее чем 150 заявок.

3. Математические ожидания и среднеквадратичные отклонения для времени генерации очередной заявки и обслуживания по каждой бензоколонке должны быть настраиваемыми, также как и предельное количество заявок в очереди. Емкость очереди должна быть подобрана так, чтобы не происходило потерь заявок.

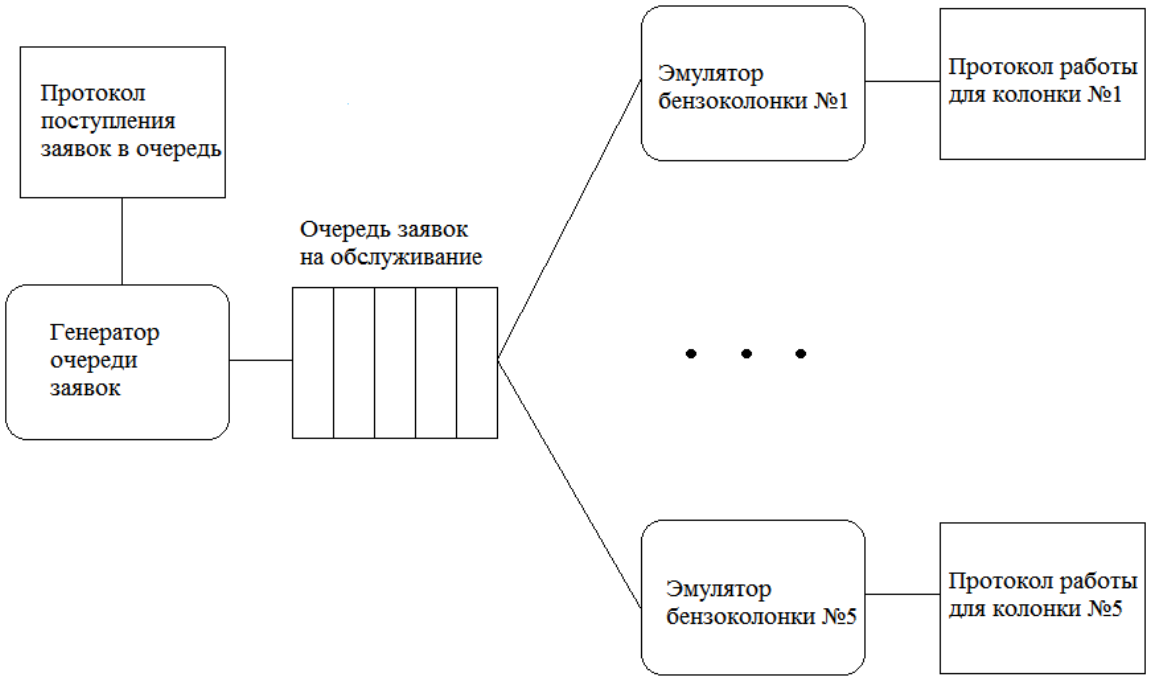

Рисунок 2.- Модель задачи из лабораторной работы №1.

4. Исходную задачу необходимо решить с использованием семафоров и разделяемой памяти (с использованием функций ftok, semget, semop, semctl, shmget, shmat, shmdt, shmctl). Описания необходимых функций приведены во встроенной справочной системе UNIX/Linux и в [1].