быть не меньше чем заявляемая минимальная нагрузка на преподавателя.

По результатам формирования примерных потоков преподавателями система генерирует учебные потоки для последующей регистрации на дисциплины студентами. Все недостающие обязательные потоки создаются руководством кафедры. Заведующий кафедрой отслеживает процесс формирования индивидуальных учебных нагрузок и контролирует, чтобы студенты имели возможность выбрать не только дисциплины обязательной и вариативной части, но и альтернативные.

На следующем этапе открывается регистрация студентов на дисциплины. Студент, в соответствии с финансовыми возможностями, с учетом требований по максимальной и минимальной учебной нагрузке, составляет индивидуальный учебный план на следующий учебный год. Студент знакомится с каталогом дисциплин: смотрит описание, формируемые компетенции, историю модификации курса, статистику по обучению и успеваемости, а также отзывы завершивших изучение курса. Выбор дисциплин регулируется согласно расставленным пререквизитам. Обучающийся имеет право изучить дисциплину в обход требований по пререквизитам, если сможет подтвердить набор знаний и/или компетенций, необходимых для изучения выбранного курса. Решение принимается специальной комиссией профильной кафедры. Для составления рационального и оптимального индивидуального учебного плана обучающийся может воспользоваться помощью консультанта.

Вместе с учебной нагрузкой, студенту должен быть предоставлен финансовый калькулятор, когда он может оценить примерную сумму, которую ему необходимо будет оплатить за каждый планируемый учебный курс на следующий учебный год (с учетом возможного рефинансирования). Студентам, обучающимся на бюджетной основе, также будет показана примерная стипендия, которую они будут получать за каждый учебный семестр. Такая схема дает студенту возможность контролировать не только свою нагрузку, но и смотреть на финансовое обеспечение обучения.

После завершения периода регистрации на дисциплины, запускается процесс проверки учебных потоков на качество, по результатам работы которой, вносятся корректировки в учебную нагрузку, нерентабельные потоки расформировываются (и уничтожаются). Тьюторы отрабатывают с каждым учащимся, не завершившим удачно процесс регистрации на дисциплины индивидуальную траекторию обучения.

По окончании процесса регистрации (перерегистрации) на дисциплины, формируются, печатаются и подписываются регистрационные формы на каждого обучающегося. На основе регистрационной формы формируется индивидуальный учебный план студента.

По итогам регистрации (перерегистрации) на дисциплины формируется расчет часов по кафедрам. Руководство кафедры распределяет свободную учебную нагрузку между преподавателями. На основе распределенного расчета часов формируется поручение учебной нагрузки по кафедре, а также университетский расчет часов.

**Заключение**

Индивидуальная траектория обучения вносит корректировки в организацию учебного процесса: теперь не нужно формировать приказ для перевода студента с курса на курс. При формировании индивидуальной траектории обучения, при откладывании изучения некоторых дисциплин на другой срок, студент видит в реальном времени, как меняется его индивидуальный учебный план: изменяется примерный срок обучения, учебная нагрузка на каждый год, а также может управлять финансовым вопросом.

Таким образом, в университете формируется банк дисциплин и слушателей, а также появляется возможность объективной оценки соответствия содержания учебных программ требованиям работодателей.

# ПОВЫШЕНИЕ ОТКАЗОУСТОЙЧИВОСТИ МНОГОПРОЦЕССНОЙ ПРОГРАММНОЙ СИСТЕМЫ ПУТЕМ МОДЕРНИЗАЦИИ МЕХАНИЗМА МЕЖПРОЦЕССНОГО ВЗАИМОДЕЙСТВИЯ

Мейта Р.В.
Научный руководитель: Шамин А.А., доцент кафедры ИПС
Томский политехнический университет
634050, Россия, г. Томск, пр-т Ленина, 30
E-mail: theshrodingerscat@gmail.com

Предметом обсуждения в данной работе являются POSIX-совместимые механизмы обеспечения межпроцессного взаимодействия в семействе операционных систем GNU/Linux.

**Введение**

Под межпроцессным взаимодействием (или IPC – Inter-Process Communication) в целом понимают набор способов обмена данными между множеством потоков в одном или более процессах. Процесс - это находящаяся в состоянии выполнения программа вместе со средой ее выпол-

нения. Процессы могут быть запущены на одном или более компьютерах, связанных между собой сетью.

В этой работе рассматривается проблема выбора оптимальной технологии IPC для мультипроцессного программного комплекса в контексте повышении отказоустойчивости при наличии возможности незапланированного завершения одного или нескольких процессов системы. Рассматриваемая программная система является частью программного обеспечения встраиваемого вычислительного узла, основными задачами которого является геолокация и передача данных. Встраиваемый характер аппаратной части налагает определенные ограничения на программный комплекс, частным случаем которых является небольшое количество оперативной памяти, доступной устройству. На программном уровне, вычислительный узел находится под управлением операционной системы семейства GNU/Linux, которая, в случае возникновения ситуации нехватки памяти, может принять решение об отключении части выполняющихся в данный момент процессов. Задача восстановления работы отдельных модулей рассматриваемой программной системы может быть решена путем обработки POSIX-сигналов, однако восстановление коммуникаций между процессами связано с определенными трудностями.

**Общая схема работы программного комплекса**

Архитектура изучаемой программной системы представлена комплексной топологией типа «звезда» (рис. 1). Программный комплекс состоит из N самостоятельных программ, N-1 из которых ориентированы на выполнение своих узкоспециализированных задач, и одного процесса-администратора. В число основных функций процесса-администратора, или корневого процесса, входит контроль за состоянием остальных процессов системы, а так же диспетчеризация сообщений между процессами.

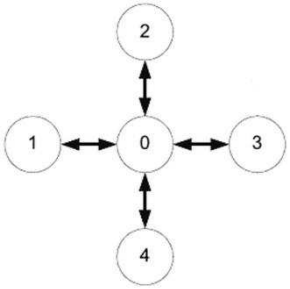


Рис. 1. Топология процессов типа «звезда»

**Именованные каналы**

Именованный канал (pipe) является однонаправленным коммуникационным мостом между двумя процессами и может использоваться для поддержки и контроля информационного потока [1]. Традиционный канал – «безымянен», поскольку существует анонимно и только во время выполнения процесса. Именованный канал может принимать только определенный объем данных (обычно 4 Кб). Если он заполнен, процесс останавливается до тех пор, пока хотя бы один байт не будет прочитан и не появится свободное место, чтобы снова заполнить его данными. С другой стороны, если канал пуст, то читающий процесс останавливается до тех пор, пока пишущий процесс не внесёт данные. Условно именованный канал можно представить в виде трубы (рис. 2).

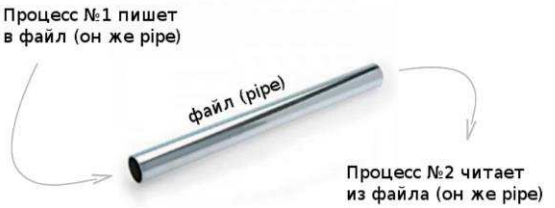


Рис. 2. Схематическое представление именованного канала

Изначально в рассматриваемой программной системе использовалась именно такая технология IPC. Таким образом, изначальная топология коммуникаций совпадала с общей архитектурой системы, так как центральным узлом соединения всех именованных каналов выступал процесс-администратор. Однако, использование именованных каналов имеет существенные недостатки. К их числу можно отнести необходимость порождения всех прикладных процессов от процесса-администратора из-за необходимости передачи дескриптора канала при его создании, а так же генерацию сигнала обрыва канала при завершении одного из процессов-потомков. Такая структура приводит к невозможности отдельного запуска прикладного процесса, что значительно снижает надежность программной системы.

**Разделяемая память**

Альтернативой использования именованных каналов является механизм разделяемой памяти. Разделяемую память применяют для того, чтобы увеличить скорость прохождения данных между процессами. Техника разделяемой памяти позволяет осуществить обмен информацией не через ядро (как это делают именованные каналы), а используя некоторую часть виртуального адресного пространства, куда помещаются и откуда считываются данные [2]. Схематично она может быть изображена как некая именованная область в памяти, к которой обращаются одновременно множество процессов (рис. 3).

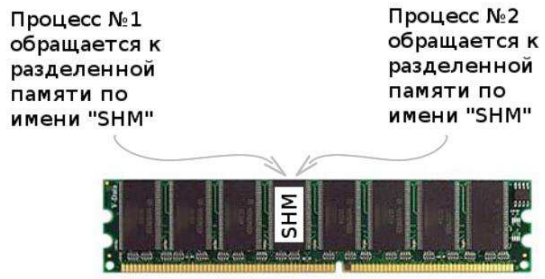


Рис. 3. Схематическое представление разделяемой памяти

После создания разделяемого сегмента памяти любой из процессов может подсоединить его к своему собственному виртуальному пространству и работать с ним, как с обычным сегментом памяти. Существенными недостатками такого обмена информацией является отсутствие каких бы то ни было средств синхронизации, что является очень критичным при большом числе процессов, и отсутствие заданной структуры сообщений.

**Семафоры и мьютексы**

Семафор – самый часто употребляемый метод для синхронизации потоков и для контролирования одновременного доступа множеством потоков/процессов к общей памяти (к примеру, глобальной переменной) [3]. Взаимодействие между процессами в случае с семафорами заключается в том, что процессы работают с одним и тем же набором данных и корректируют свое поведение в зависимости от этих данных. Существую два типа семафоров:

- семафор со счетчиком, определяющий лимит ресурсов для процессов, получающих доступ к ним;
- бинарный семафор (или мьютекс), имеющий два состояния «0» или «1» (чаще: «занят» или «не занят»).

Однако полноценная коммуникация в межпроцессной системе не может быть организована при помощи этой технологии IPC.

**Очереди сообщений**

Очереди сообщений как средство межпроцессной связи дают возможность процессам взаимодействовать, обмениваясь данными. Данные передаются между процессами дискретными порциями, называемыми сообщениями. Процессы, использующие этот тип межпроцессной связи, могут выполнять две операции: послать сообщение и принять сообщение. После создания процесс должен зарегистрироваться на общей коммуникационной шине (очереди сообщений). Одной из реализаций технологии очереди сообщений является открытая кроссплатформенная библиотека ZeroMQ, которая была использована в данной работе.

**Заключение**

Таким образом, наиболее подходящим кандидатом для замены применяемого изначально подхода к организации межпроцессного взаимодействия в рассматриваемой программной системе является механизм очереди сообщений. Объект очереди существует независимо от процессов системы, таким образом обеспечивая необходимую степень отказоустойчивости системы, так как завершение любого из процессов не приведет к обрыву канала связи. Кроме этого, процессы могут быть запущены независимо друг от друга и в разные моменты времени. Использование механизма очереди сообщения позволило уменьшить связность модулей системы, а так же сократить информационную нагрузку на корневой процесс. При наличии единой шины связи между процессами, процесс-администратор перестал являться центром коммуникации системы, и используется только для контроля состояния системы в целом. Сообщения могут передаваться от любого процесса любому (рис. 4).

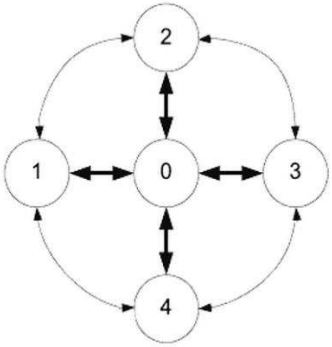


Рис. 4. Обновленная топология взаимодействия процессов

**Литература**

1. Мэтью Н., Стоунс Р. Основы программирования в Linux. – СПб.: БХВ-Петербург, 2009. – 896 с.

2. Лав Р. Linux. Системное программирование. – СПб.: Питер, 2008. – 416 с.

3. Стивенс У. Unix. Взаимодействие процессов. – СПб.: Питер, 2002. – 574 с.