## Введение

В этой статье рассматривается алгоритм синхронизации времени внутри секунды, его назначение, а так же приводится описание существующего в ядре Linux алгоритма и сложности, возникающие при его реализации.

Рассмотрим следующую систему: имеется машина с установленной Linux-based операционной системой и различными подключенными внешними устройствами. На машине запущены одна или несколько программ, которым для работы требуется, чтобы системные часы шли согласно некоторому эталонному источнику времени. Например, такой программой может быть драйвер сетевого адаптера, который должен регистрировать приходящие в систему извне пакеты данных и ставить в соответствие каждому из них некоторую метку времени, а источником могут быть часы другой системы, или одно из устройств, подключенных к рассматриваемой системе.

То есть, таким программам требуется, чтобы системные часы были *синхронизированы* с эталонным источником. Это означает, что в каждый момент времени значения часов источника и того, кто с ним синхронизируется, должны как можно меньше отличатся. Чем меньше разница, тем выше точность синхронизации.

Источников, как и программ, нуждающихся в синхронизации, может быть несколько. Для синхронизации значений часов системы может быть использован NTP (Network Time Protocol), дающий относительно неплохую точность синхронизации, порядка миллисекунд. Но иногда такой точности недостаточно, и тогда помогают алгоритмы, точность которых достигает порядка наносекунд.

Рассматриваемый далее в статье алгоритм позволяет с помощью некоторого высокоточного генератора PPS (Pulse per second) сигналов узнавать о начале каждой новой секунды и сообщать об этом системе, в которой, в свою очередь, регистрируются приложения - PPS-потребители, которым требуется реагировать на эти сигналы. Каждый PPS-сигнал вызывает в системе рассылку текущего значения системных часов всем зарегистрированным потребителям.

То есть для получения высокой точности можно использовать какой-нибудь относительно грубый алгоритм, вроде NTP, в совокупности с PPS. И тогда если время будет синхронизироваться с помощью основного алгоритма до хотя бы секунд, то уже внутри секунды синхронизацию обеспечит PPS.

Рассмотрим далее более конкретно, что такое метки времени, откуда берется PPS-сигнал, и что это такое, какие сложности возникают при реализации алгоритмов, работающих с PPS, а так же конкретный пример - алгоритм, использующийся в ядре Linux для синхронизации системного времени с PPS-источником.

## Время в ядре Linux

В компьютере время представлено как значение некоторого постоянно возрастающего аппаратного счетчика, значение которого увеличивается через примерно равные промежутки времени. Счетчик не может инкрементироваться через просто равные промежутки времени, так как частота его инкрементации связана с частотой некоторого генератора, а его частота колеблется в относительно небольшом диапазоне, зависящем от условий, в которых находится генератор, например, от давления и температуры, от энергопотребления и других причин.

Значение этого счетчика в каждый момент времени - целое число, которое используется системой для перевода его в человеческие единицы измерения времени. Говоря о человеческих единицах времени будем здесь и далее иметь ввиду наносекунды. Для перевода значения счетчика в наносекунды требуется знать, с примерно какой частотой счетчик инкрементируется - для каждой системы устройств эта частота известна заранее. Зная только это значение счетчика, мы можем получить так называемое *monotonic time -* "сырое" значение времени, к которому не применялись никакие алгоритмы для его коррекции. Так же в системе есть понятие *real time -* это время, получаемое из monotonic с помощью применения различных алгоритмов синхронизации времени - это то время, которое доступно в пользовательской области операционной системы для приложений вне ядра, и которое здесь, в статье, называется *системными часами*.

Когда говорят о коррекции времени, то имеют ввиду не прямое изменение значения аппаратного счетчика, а способ его преобразования в real time из monotonic time. Для этого используется множество различных коэффициентов, значения которых вычисляются алгоритмами синхронизации.

Чтобы понять, как инкрементируется значение аппаратного счетчика, надо понять, что такое прерывания. Прерывание - это сигнал, сообщающий процессору о наступлении какого-либо события, управление при возникновении прерывания передается его обработчику. Прерывание может быть как аппаратным, в случае счетчика времени, так и программным, если оно инициировано специальной инструкцией в коде.

То есть в системе через примерно равные промежутки времени происходят аппаратные прерывания, по которым увеличивается аппаратный счетчик времени. Выше мы рассматривали систему, часы которой надо синхронизировать, и систему или устройство, с которым это надо сделать. У каждого из рассматриваемых объектов есть свои счетчики времени, значения которых мы менять не можем, но можно менять коэффициенты, по которым получается real time из monotonic time. Вычислением этих коэффициентов и занимается рассматриваемый алгоритм синхронизации, использующий PPS.

## Pulse per second

Pulse per second (PPS) - сигнал, генерируемый некоторым высокоточным источником (PPS-source) с частотой максимально близкой к одной секунде. Для рассматриваемого в статье алгоритма важно уметь "ловить" моменты, когда возникает PPS-сигнал, то есть получать метки времени точно в эти моменты.

PPS-источник подключается к вычислительной системе посредством контакта интерфейса последовательной линии (modem-control pin on a serial-line interface) или, чаще, параллельной . Система узнает о возникновении сигнала, получая прерывание, и как можно скорее запоминает текущую метку времени. Далее эта метка становится доступна с относительно небольшой задержкой после ее получения для ПО, работающего со временем - PPS-потребителей, которые ранее зарегистрировались в системе, как получатели PPS-сигналов. С помощью этой метки можно, например, получить расхождение между источником точного времени и системными часами.

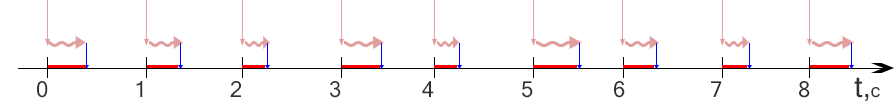
К одной системе может быть подключено несколько PPS-источников, каждый из которых генерирует собственные PPS-сигналы независимо от других. Когда очередной PPS-потребитель регистрируется на прием сигналов, он может указать, какой PPS-источник ему нужно слушать, причем слушать можно несколько источников.

Кроме пользовательских программ в ядре есть еще один потребитель сигналов: специальная процедура ядра - *hardpps*. Эта процедура на основании получаемых сигналов пересчитывает коэффициенты, по которым monotonic time пересчитывается в real time, в результате чего изменяется системное время. Причем эта процедура может слушать единовременно только один PPS-источник. Как раз с ним рассматриваемая система и будет синхронизировать свои часы.

Рассмотрим, какие возникают сложности при реализации алгоритмов синхронизации, основанных на PPS. Первая задача, требующая решения, это как правильно учесть время, которое пройдет с момента генерации сигнала и возникновения прерывания до вызова его обработчика и запоминания текущего времени для его передачи PPS-потребителям. Во-вторых, что ставить как метку времени для PPS-сигнала? В-третьих, как использовать получаемые метки для коррекции системного времени в процедуре hardpps, описанной выше?

Вторая задача - это выбор между меткой real time и monotonic time. Эту задачу решает API ядра, предоставляющий программисту самому выбрать, какая метка ему нужна - каждому сигналу ставятся в соответствие сразу две метки - и real time, и monotonic time, так же называемый raw time.

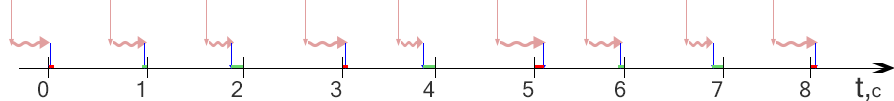
Третья задача будет рассмотрена далее при описании работы процедуры hardpps. Рассмотрим, в чем суть первой задачи, с помощью следующего рисунка:



На рисунке изображена шкала времени по секундам, где каждое деление на шкале обозначает возникновение сигнала PPS. Самые высокие стрелки - это момент, когда мы понимаем, что прошел сигнал PPS; волнистые стрелки - это время, которое тратится на обработку прерывания, и, наконец, синие стрелки - это момент, когда мы готовы прочитать метку времени, чтобы поставить ее в соответствие возникшему сигналу. Как видно из графика - если обработчик прерывания "очнется" только в момент возникновения сигнала, то метка времени, которая будет считана к нему, будет получена с отставанием, причем каждый раз с разным, как видно из красных полос на рисунке.

Такое неравенство отставаний получается в силу того, что не всегда за одно и то же время управление передается обработчику прерывания, так как система каждый раз загружена по-разному, и планировщик каждый раз ставит выполнение обработчика на разные позиции.

В качестве решения алгоритм, рассматриваемый в статье, начинает ожидать сигнал немного раньше, чем он возникнет. Это делается путем оценки того, сколько примерно может занять времени получение метки после возникновения сигнала. И тогда если начать ожидать сигнал чуть раньше, то к тому моменту, как он появится, мы будем готовы сразу считать текущую метку времени. Описанный способ хорошо видно на следующем рисунке:

Здесь мы видим, что если подготовится к сигналу пораньше, то почти всегда метка будет считана достаточно точно. Здесь зеленые полосы - это активное ожидание прихода сигнала.

## Список литературы

1. Cochran R., Marinescu C., Riesch C. Synchronizing the Linux system time to a PTP hardware clock //Precision Clock Synchronization for Measurement Control and Communication (ISPCS), 2011 International IEEE Symposium on. – IEEE, 2011. – С. 87-92.
2. Mogul J. C. et al. Pulse-per-second api for unix-like operating systems, version 1.0 //Pulse. – 2000.
3. Gleixner T., Niehaus D. Hrtimers and beyond: Transforming the linux time subsystems //Proceedings of the Linux symposium. – 2006. – Т. 1. – С. 333-346.
4. Köker K., Hielscher K. S., German R. A Low-Cost High Precision Time Measurement Infrastructure for Embedded Mobile Systems //Robot Motion and Control 2007. – Springer London, 2007. – С. 445-452.