## 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.

Рассматриваемый 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, которым требуется реагировать на эти сигналы. Каждый PPS-сигнал инициирует прерывание в ядре, по которому PPS-потребители (consumers) получают метки времени, взятые в момент этого прерывания.

Эти метки могут быть использованы приложениями каждым в отдельности, или с их помощью можно корректировать общее системное время так, что это отразится на всех приложениях.

Сами по себе метки могут быть нужны, например, если в системе есть некоторое устройство, пусть это будет сетевой адаптер, который регистрирует события, происходящие вне системы, и он должен ставить в соответствие каждому событию метку времени, причем как можно точнее, и если абсолютную величину временной метки можно синхронизировать с некоторым эталонным временем более грубыми алгоритмами в пределах милли- или микросекунд, то внутри каждой секунды задача синхронизации может решаться отдельно для достижения еще более высокой точности, вплоть до наносекунд.

Описав общий принцип работы алгоритма, рассмотрим далее более конкретно, что такое прерывания, метки времени, откуда берется PPS-сигнал, и что это такое, и какие сложности возникают при реализации такого алгоритма.

## 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.

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.

Когда говорят о коррекции времени, то имеют ввиду не прямое изменение значения аппаратного счетчика, а способ его преобразования в real 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. Далее следует обзор PPS.

## Pulse per second

Pulse per second (PPS) - сигнал, генерируемый некоторым высокоточным источником (PPS-source) с частотой максимально близкой к одной секунде.

PPS-источник подключается к вычислительной системе посредством контакта интерфейса последовательной линии (modem-control pin on a serial-line interface) . 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-consumers. С помощью этой метки можно, например, получить расхождение между источником точного времени и системными часами.

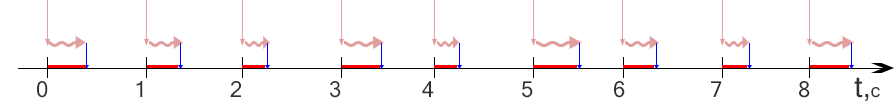
Так же эта метка может быть передана процедуре ядра - *hardpps,* которая используется для коррекции системного времени и будет рассмотрена далее.

К одной системе может быть подключено несколько PPS-источников, и пользовательские программы могут использовать их для получения PPS-сигналов, но для коррекции системного времени может использоваться только один PPS-источник единовременно.

Синхронизация посредством PPS предполагает его использование совместно с такими алгоритмами, как NTP (Network time protocol).

Рассмотрим, какие возникают сложности при реализации 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-получателям. Во-вторых, что ставить как метку времени для PPS-сигнала? В-третьих, как использовать получаемые метки для коррекции системного времени в процедуре hardpps?

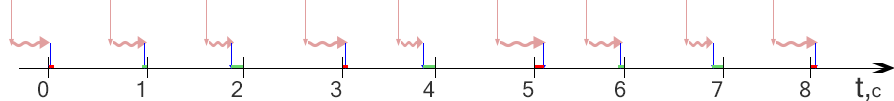
Вторая задача - это выбор между меткой real time и monotonic time. Эту задачу решает API ядра, предоставляющий программисту самому выбрать, какая метка ему нужна - каждому сигналу ставятся в соответствие сразу две метки - и real time, и monotonic time, так же называемый raw time.

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 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 планировщик каждый раз загружен по-разному.

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.