

# FreeBSD 闰秒的支持

## 目录

1. 介绍.....	1
2. FreeBSD 闰秒的默认处理方式 .....	1
3. 注意.....	1
4. 闰秒.....	2
5. 闰秒.....	2

## 1. 介绍

闰秒是为了同地球自转，而原子钟标准所做的特定修正。本文描述了 FreeBSD 如何处理闰秒。

截至本文完稿时，下一个闰秒将会发生在2015年6月30日23:59:60 UTC。这个闰秒将会发生在南北美洲和亚太地区的一个工作日里。

闰秒是由 IERS 在 [Bulletin C](#) 上宣布的。

[RFC 7164](#) 描述了闰秒的准行。也可参 [time2posix\(3\)](#)。

## 2. FreeBSD 闰秒的默认处理方式

处理闰秒最可靠的方法是使用 FreeBSD 的 POSIX 时钟，以及 [NTP](#)。如果 [ntpd\(8\)](#) 正在运行，并且正确和正处理闰秒的上游 NTP 服务器同步，闰秒将使系统自重新当天的最后一秒。不需要进行其它调整。

如果上游的 NTP 服务器没有正确处理闰秒，[ntpd\(8\)](#) 会在它的上游服务器同步并修正后，跟着加上一秒。

如果未使用 NTP，将需要在闰秒后手动调整系统。

## 3. 注意

闰秒在全世界的同一瞬间入：UTC 午夜。日本在上午，太平洋在正午，美洲在傍晚，而欧洲在晚上。

我相信并期望，如果提供了正确和稳定的 NTP 服务，FreeBSD 会在闰秒按操作，正如在之前遇到闰秒一样。

然而我要警告，事实上没有应用程序会向内核关于闰秒的事。我的意思是，闰秒正如它的一秒，本上是闰秒前一秒的重播，而大部分应用程序作者则是意想不到的事。

其它操作系统和它们可能会也可能不会像 FreeBSD 一样处理闰秒，没有正确和稳定 NTP 服务的系统一点也不知道闰秒的诞生。

闰秒因闰秒而崩溃并非前所未闻，显示，大量的公共 NTP 服务器可能会管理和公告闰秒。

即使不会因秒而产生任何可怕的事情。

## 4. 秒

是否将使用秒是可行的。由于 NTP 的特性，可能要行到秒前24小时。有些主要的参考来源只在秒事件前一小公告。NTP 守护进程：

```
% ntpq -c 'rv 0 leap'
```

包含 `leap_add_sec` 的输出表明了于秒的正支持。`leap_none` 会在秒前24小时或秒后示。

## 5. 秒

在实践中，FreeBSD 中的秒通常不是个问题。我希望篇文章能解清楚方面可能出的状况，以及如何使秒事件行得更利。