linux认证辅导:perl的时间处理函数Linux认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/619/2021_2022_linux_E8_AE_A4_E8_AF_c103_619625.htm 表示日期的方式多种多样：“ 18 Jan 1973 " ， ” 18/01/1973 " ， ” 01/18/1973 " ， ” Jan 18 1973 " ， ” 18-01-73 " ， ” 18-01-1973 " ， ” 01/73 "，其中一些格式含义不清(如 ” 01-06-1973 " 是表示6月1日呢，还是表示1月6日呢？) 如果不规定日期的表示形式，是很难处理的。 想了解 ” 18 Jan 1973 " 和 ” 6 Sep 1950 " 之间的区别，需要把它们转换为数字表示。Unix 内部使用纪元秒表示时间。日期和时间加起来表示之自格林威志时间1970年1月1日午夜时分(纪元)到当前时刻之间的秒数。 ” 18 Jan 1973 " (假定为午夜时分) 的纪元秒为96163200。在该系统中，午夜表示一天的开始时刻。 让我们生成一个日期通过Perl中提供的gmtime函数，你可以自己来验证这点。给定一个用以表示自从纪元以来的秒数的整数，通过gmtime函数可以计算出代表相应的日期和时刻，例如： perl -le ‘ print scalar gmtime 96163200 ’ Thu Jan 18 00:00:00 1973 调用 gmtime() 函数，你会得到一系列值的列表，包括时，分，秒，日期，月份，年份等等。 perl -le ‘ print join(” , ” , gmtime 96163200) ’ 0,0,0,18,0,73,4,17,0 前面3个0分别表示秒，分，时。小时是从0-23，故下午是12时往后。第4个数表示该月中的天数(本例中为18号)。第5个数表示月份，从0开始(代表1月份)。之所以从0开始，是因为月份对应着月份数组的下标： @months = qw(Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec). $month = (gmtime 96163200)[4]. # “ Jan ” 年份 (本例中为73)的表示有点特殊。它并不是年份的最后两位数字

。它表示从1900年开始的年份。为什么要这样表示呢？这是因为 C语言就是这样处理的。Perl试图使得其库和系统调用尽量接近操作系统的处理方式。 所以，如果你想输出4位数的年份，表示如下： $year = (gmtime 96163200)[5] 1900. 如果你不了解这种处理方式，就会制造出Y2K问题，你也许会这样写：$year =“ 19 ″ . (gmtime 96163200)[5]. # 出错！ 2000年将变为19100 对于gmtime()函数的返回值还没有介绍完，还有4, 17, 和 0这3个数。它们分别表示一星期中的第几天 (星期日为0), 一年中的第几天(0 表示一年中的第一天), 以及是否采用夏时制(表示不采用，正数表示采用，负数表示不可知)。 Perl中的time() 函数返回以纪元秒形式表示的当前日期和时间。如果你打算把它转换为字符串，就可使用gmtime() 和localtime() 函数： $now = localtime(time()). ($sec, $min, $hour, $day, $mon, $year, $wday, $yday, $isdst) = localtime(time()). 如果调用 localtime() 或gmtime() 时不带参数，它将自己调用time() $now = localtime(). ($sec, $min, $hour, $day, $mon, $year, $wday, $yday, $isdst) = localtime(). 常见的日期和时间操作 如果你打算计算两个时刻之间的时间段，只需将它们转换为相应的纪元秒，然后两数相减即可： $difference_in_seconds = $later_datetime -$earlier_datetime. 要把秒转换为分，时，或天数，只需要分别将它们除以60, 3600 和 86400 即可： $difference_in_minutes = $difference_in_seconds / 60. $difference_in_hours = $difference_in_seconds / 3600. $difference_in_day = $difference_in_seconds / 86400. 反过来做，你也可以回答如下问题：”4天后是几号？”： $then = time() 86400 * 4. print scalar localtime $then. 它给出的答案精确到秒。例如，如果4天

后的纪元秒值为932836935, 你可以输出日期的字符串如下；Sat Jul 24 11:23:17 1999 如果你打算输出那个日期的午夜时分 (如 ” Sat Jul 24 00:00:00 1999 ") 使用如下模块：$then = $then - $then % 86400. # 去掉那个日期的尾巴 类似地，你可以用四舍五入法，输出最靠近午夜时分的日期： $then = 43200. # add on half a day $then = $then - $then % 86400.# truncate to the day 如果你的时区距离GMT为相差偶数个小时，这就管用了。并不是所有的时区都是很容易处理的。你所真正需要的是在你自己的时区内计算纪元秒，而不是在GMT中计算。 Perl 中的名为Time::Local的模块，可以提供两个函数 timelocal() 和timegm()。其返回值同 localtime() 和gmtime() 一样。 use Time::Local. $then = time() 4*86400. $then = timegm localtime $then. # local epoch seconds $then -= $then % 86400. # truncate to the day $then = timelocal gmtime $then. # back to gmt epoch seconds print scalar localtime $then, “ \n ”. 日常生活所用的日期和时间的表示 你已经级掌握了时，分，年等值的含义，也了解了纪元秒的含义。而日常生活中的日期和时间是用字符串来表示的，你怎样才能把日常所用的日期和时间串格式转换成纪元秒呢？ 方法之一是写出语法分析小程序，该方法灵活而快速： use Time::Local. @months{qw(Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec)} = (0..11). $_ = “ 19 Dec 1997 15:30:02 " . /(\d\d)\s (\w)\s (\d)\s (\d):(\d):(\d)/ or die “ Not a date ”. $mday = $1. $mon = exists($months{$2}) ? $months{$2} : die “ Bad month ”. $year = $3 - 1900. ($h, $m, $s) = ($4, $5, $6). $epoch_seconds = timelocal($s,$m,$h,$mday,$mon,$year). 一个更通用些的方法，是从CPAN安装Date::Manip 模块。 use

Date::Manip. $epoch_seconds = UnixDate(” 19 Dec 1997 15:30:02 ″ , ” s ”). 注意，由于 Date::Manip是个大模块，使用该模块时，将会增加你的程序的启动时间。其中一个原因是 Date::Manip将对多种不同的格式进行识别，如：“ today ” “ now ” “ first sunday in april 2000 ″ “ 3:15, today ” “ 3:15pm, first sunday in april 2000 ″ “ 2000/01/18 09:15 ″ Date Manipulation 2036, 2037, 2038, ⋯, 1901?! 大多数C程序把纪元秒存为有符号整数，可表示正的和负的日期，但计算机存储器所表示的整数大linux认证更多详细资料小是有限的，用有限的位数来表示秒。这就是说，我们在计算纪元秒时，所表示的日期是有限制的。 确切的限度取决于你的机器所能表示的整数的位数。 Perl最多以32位的长度存储整数。粗略地讲，有一位用来表示正负号，其余31位来表示数。如果8位，你可以存储的最大数是255，即2的8次方减1。故Perl中所存储的32位符号数中的最大数为： print 2**31-1, “ \n ” . 2147483647 这个数字对应了哪个日期呢？ print scalar(gmtime 2**31-1), “ \n ” . Tue Jan 19 03:14:07 2038 在那个时刻的1秒之后会发生什么呢？ print scalar(gmtime 2**31), “ \n ” . Fri Dec 13 20:45:52 1901 啊！发生了什么？对于32位有符号整数来说， 2**31太大了。它 ” 翻卷过去了 ” ，其符号位被置为负号，因而成为了所能表示的最大负数。这对应于1970年开始时刻之前的秒的最大值。 其结果说明了什么呢？你不能存储gmtime(2**31)之前或gmtime(2**31-1)之后的以纪元秒表示的日期。 你可千万不要想不开，这可不是什么大问题。如果你要用到32位有符号整数表示的纪元秒以外的时间，你只需要改变你的表示方式，你可从CPAN中找到不少日期模块，其中的Date::Calc

和Date::Manip 很可能是功能最强的两个模块。 这两个模块使用自己的日期表示方式，以避免Y1901-Y2038 的限制。Date::Manip 使用罗马历法，从公元 0000 到公元9999。Date::Calc 也使用罗马历法，可表示的年份从1 到32767。 总结 对于在1902-2037范围内的日期和时期表示, 把它们转换为纪元秒，要存取这些数，你只需使用整数算术运算，gmtime() 和 localtime()函数，以及标准的Time::Local模块。如果要对该范围以外的日期进行计算或者要分析某特殊的日期格式，你可以使用CPAN 中的Date::Manip 和Date::Calc模块。