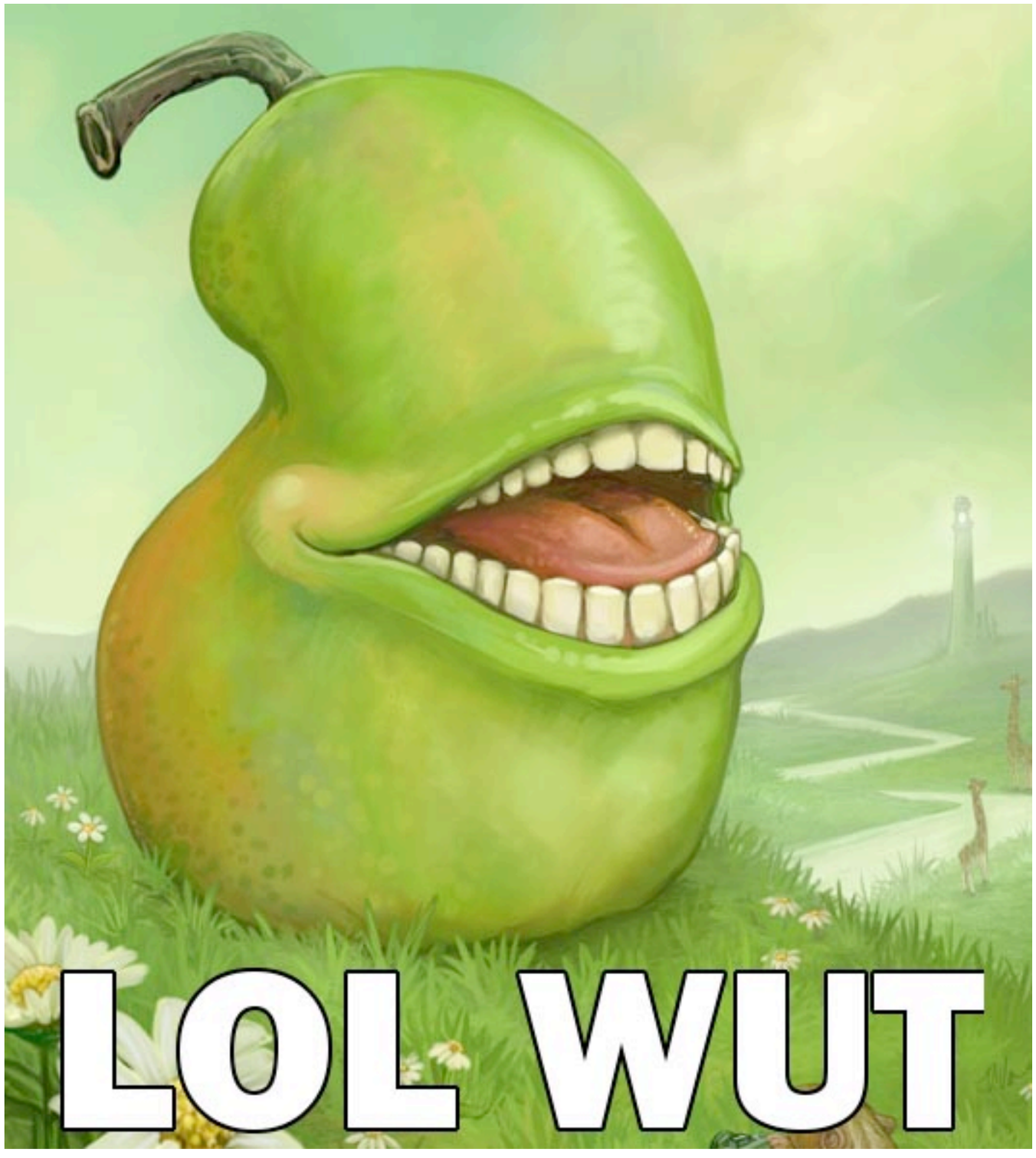
# What's the time, Mr Wolf?

limitations & workarounds
for rfc3164 timestamps


LOL WUT

# I see no problem here.

```
Jan 23 01:45:13 quasimodo postfix/smtpd[3578]: disconnect ↵
        from kmr04-home.tm.net.my[218.111.184.22]
```

# oh wait.

```
Jan 23 01:45:13 quasimodo postfix/smtpd[3578]: disconnect ↵
    from kmr04-home.tm.net.my[218.111.184.22]
```

wtf happened to my year?

```
Jan 23 01:45:13 quasimodo postfix/smtpd[3578]: disconnect from ↵
                kmr04-home.tm.net.my[218.111.184.22]
Jan 23 01:45:13 quasimodo postfix/smtpd[27824]: lost connection ↵
                after RCPT from unknown[87.19.209.206]
```

# 2008

```
Jan 23 01:45:13 quasimodo postfix/smtpd[3578]: disconnect from ↵
                kmr04-home.tm.net.my[218.111.184.22]
```

```
Jan 23 01:45:13 quasimodo postfix/smtpd[27824]: lost connection ↵
                after RCPT from unknown[87.19.209.206]
```

# 2006

```
Oct 23 01:48:16 snowballs sshd[3769]: (pam_unix) session ↵
opened for user harry by (uid=0)
```

# HEADER

```
Oct 23 01:48:16 snowballs sshd[3769]: (pam_unix) session ↵
        opened for user harry by (uid=0)
```

Mmm dd hh:mm:ss

# HEADER

hostname|IPV4|IPV6

Oct 23 01:48:16 snowballs sshd[3769]: (pam_unix) session ↵
opened for user harry by (uid=0)

Mmm dd hh:mm:ss

# MSG

process name

```
Oct 23 01:48:16 snowballs sshd[3769]: (pam_unix) session ↵
opened for user harry by (uid=0)
```

# MSG

process name

```
Oct 23 01:48:16 snowballs sshd[3769]: (pam_unix) session ↵
opened for user harry by (uid=0)
```

message

# Section 5.1

## 5.1 Dates and Times

It has been found that some network administrators like to archive their syslog messages over long periods of time. It has been seen that some original syslog messages contain a more explicit time stamp in which a 2 character or 4 character year field immediately follows the space terminating the TIMESTAMP. This is not consistent with the original intent of the order and format of the fields. If implementers wish to contain a more specific date and time stamp within the transmitted message, it should be within the CONTENT field. Implementers may wish to utilize the ISO 8601 [7] date and time formats if they want to include more explicit date and time information.

Additional methods to address this desire for long-term archiving have been proposed and some have been successfully implemented. One such method is that the network administrators may choose to modify the messages stored on their collectors. They may run a simple script to add the year, and any other information, to each stored record. Alternatively, the script may replace the stored time with a format more appropriate for the needs of the network administrators. Another alternative has been to insert a record into the file that contains the current year. By association then, all other records near that informative record should have been received in that same year. Neither of these however, addresses the issue of associating a correct timezone with each record.

# Section 5.1

5.1 Dates and Times

It has been found that some network administrators like to archive their syslog messages over long periods of time. It has been seen that some original syslog messages contain a more explicit time stamp in which a 2 character or 4 character year field immediately follows the space terminating the TIMESTAMP. This is not consistent with the original intent of the order and format of the fields. **If implementers wish to contain a more specific date and time stamp within the transmitted message, it should be within the CONTENT field.** Implementers may wish to utilize the ISO 8601 [7] date and time formats if they want to include more explicit date and time information.

Additional methods to address this desire for long-term archiving have been proposed and some have been successfully implemented. One such method is that the network administrators may choose to modify the messages stored on their collectors. They may **run a simple script to add the year,** and any other information, to each stored record. Alternatively, the **script may replace the stored time with a format more appropriate** for the needs of the network administrators. Another alternative has been to **insert a record into the file that contains the current year.** By association then, **all other records near that informative record should have been received in that same year.** Neither of these however, addresses the issue of associating a correct timezone with each record.

1. implementation

2. post processing

# rsyslog

"good timestamp format control; at a minimum, ISO 8601/RFC 3339 second-resolution UTC zone"

2006-11-24T20:57:50.52Z

also,

post processing

batch

```
#!/usr/bin/env ruby
#

if ARGV.size != 1 then
  puts “Usage: batch_log_reader.rb <filename>”
  exit 1
end
```

```
$ batch_log_reader messages.log
```

```
#!/usr/bin/env ruby
#

...
```

```
#!/usr/bin/env ruby
#

...

filename = ARGV[0]
```

determine file name

```
#!/usr/bin/env ruby
#

...

filename = ARGV[0]

IO.foreach(filename) do |line|
  # do something
end
```

read file in

```
#!/usr/bin/env ruby
#

...

def commit(line)
  datetime          = line[0..15]
  body              = line[16..-1]
  # store/transmit the entry
end

filename = ARGV[0]

IO.foreach(filename) do |line|
  commit(line)
end
```

process the line

```
#!/usr/bin/env ruby
#

...

def commit(line)
  datetime          = line[0..15]
  body              = line[16..-1]

  entry             = {}
  entry[:hostname] = body.split[0]
  entry[:process]  = body.split[1][0..-2]
  entry[:message]  = body.split[2..-1].join(' ')
  entry[:datetime] = datetime
  entry[:digest]   = MD5.hexdigest(line)
  # store/transmit the entry
end

filename = ARGV[0]

IO.foreach(filename) do |line|
  commit(line)
end
```

split up log entry

```
#!/usr/bin/env ruby
#

...

def commit(line)
  datetime          = line[0..15]
  body              = line[16..-1]

  entry             = {}
  entry[:hostname] = body.split[0]
  entry[:process]  = body.split[1][0..-2]
  entry[:message]  = body.split[2..-1].join(‘ ‘)
  entry[:datetime] = datetime
  entry[:digest]   = MD5.hexdigest(line)
  # store/transmit the entry
end

filename = ARGV[0]

IO.foreach(filename) do |line|
  commit(line)
end
```

```
#!/usr/bin/env ruby
#

...

def commit(line)
  datetime          = line[0..15]
  body              = line[16..-1]

  entry             = {}
  entry[:hostname] = body.split[0]
  entry[:process]  = body.split[1][0..-2]
  entry[:message]  = body.split[2..-1].join(' ')
  entry[:datetime] = rfc3164_to_ruby_datetime(datetime)
  entry[:digest]   = MD5.hexdigest(line)
  # store/transmit the entry
end

...
```

get a native time object

```
#!/usr/bin/env ruby
#

...

def rfc3164_to_ruby_datetime(timestamp)
    # magic!
end

entry[:datetime] = rfc3164_to_ruby_datetime(datetime)
```

```
#!/usr/bin/env ruby
#

...

def rfc3164_to_ruby_datetime(timestamp)
  timestamp = timestamp.split

  month = timestamp[0]
  month = Date::ABBR_MONTHNAMES.rindex(month.capitalize)
end
```

```
entry[:datetime] = rfc3164_to_ruby_datetime(datetime)
```

convert month to int

```
#!/usr/bin/env ruby
#

...

def rfc3164_to_ruby_datetime(timestamp)
  timestamp = timestamp.split

  month = timestamp[0]
  month = Date::ABBR_MONTHNAMES.rindex(month.capitalize)

  day   = timestamp[1]
  hour  = timestamp[2].split(':')[0]
  min   = timestamp[2].split(':')[1]
  sec   = timestamp[2].split(':')[2]

end

entry[:datetime] = rfc3164_to_ruby_datetime(datetime)
```

determine day, hour, min, sec

```
#!/usr/bin/env ruby
#

...

def rfc3164_to_ruby_datetime(timestamp)
  timestamp = timestamp.split

  month = timestamp[0]
  month = Date::ABBR_MONTHNAMES.rindex(month.capitalize)

  day   = timestamp[1]
  hour  = timestamp[2].split(':')[0]
  min   = timestamp[2].split(':')[1]
  sec   = timestamp[2].split(':')[2]

  year  = File.open(filename).ctime.year

  time  = Time.mktime(year, month, day, hour, min, sec)
  return time

end
```

create the time object

```
#!/usr/bin/env ruby
#

...

def rfc3164_to_ruby_datetime(timestamp)
  timestamp = timestamp.split

  month = timestamp[0]
  month = Date::ABBR_MONTHNAMES.rindex(month.capitalize)

  day   = timestamp[1]
  hour  = timestamp[2].split(':')[0]
  min   = timestamp[2].split(':')[1]
  sec   = timestamp[2].split(':')[2]

  year  = File.open(filename).ctime.year

  time  = Time.mktime(year, month, day, hour, min, sec)
  return time

end
```

```
#!/usr/bin/env ruby
#

...

def rfc3164_to_ruby_datetime(timestamp)
  timestamp = timestamp.split

  month = timestamp[0]
  month = Date::ABBR_MONTHNAMES.rindex(month.capitalize)
  day   = timestamp[1]
  hour  = timestamp[2].split(':')[0]
  min   = timestamp[2].split(':')[1]
  sec   = timestamp[2].split(':')[2]

  year  = determine_year_based_on_month(month)

  time  = Time.mktime(year, month, day, hour, min, sec)
  return time

end
```

work out the year

```
#!/usr/bin/env ruby
#

...

def determine_year_based_on_month(month)
  # last bit o' magic
end

year = determine_year_based_on_month(month)
```

```
#!/usr/bin/env ruby
#

...

def determine_year_based_on_month(month)
  if @months.last != month then
    @months << month
  end
end
```

```
year = determine_year_based_on_month(month)
```

track month

```
#!/usr/bin/env ruby
#

...

def determine_year_based_on_month(month)
  if @months.last != month then
    @months << month
    @year += 1 if month == 1
  end
  return @year
end

year = determine_year_based_on_month(month)
```

increment and/or return year

```
#!/usr/bin/env ruby
#

...

def determine_year_based_on_month(month)
  if @months.last != month then
    @months << month
    @year += 1 if month == 1
  end
  return @year
end

@months = []
@year  = File.open(filename).ctime.year

year = determine_year_based_on_month(month)
```

initialise variables

offline

```
$ stat --format="%z" messages.log
2005-02-16 18:38:39.000000000 +1100
```

```
#!/usr/bin/env ruby
#

if ARGV.size != 1 then
  puts “Usage: batch_log_reader.rb <filename>”
  exit 1
end

filename = ARGV[0]
```

```ruby
#!/usr/bin/env ruby
#

unless (1..2).member? ARGV.size then
  puts “Usage: batch_log_reader.rb <filename> [starting-year]”
  exit 1
end

filename = ARGV[0]
@year = File.open(filename).ctime.year
@year = ARGV[1] unless ARGV[1] == nil
```

```
$ batch_log_reader messages.log 2005
```

better argument handling