HOWTO zur Perl-Datei-Locking

(C) 2008-2023 T.Birnthaler/H.Gottschalk <howtos(at)ostc.de>
OSTC Open Source Training and Consulting GmbH
http://www.ostc.de

$Id: perl-file-lock-HOWTO.txt,v 1.10 2025/02/23 20:14:55 tsbirn Exp $

Dieses Dokument beschreibt die in Perl verfügbaren Operationen zum Datei-Locking.

INHALTSVERZEICHNIS

1) Einführung
2) Beispiel-Programm

---

1) Einführung

Mit "flock" wird das sogenannte "Advisory Locking" (empfohlen aber nicht erzwungen) realisiert. D.h. der konkurrierende Zugriff mehrerer Prozesse auf eine begrenzte Ressource (z.B. eine Datei) wird von diesen selber so geregelt, dass bei "exklusiven" Operationen gleichzeitig nur EIN Prozess Zugriff hat bzw. bei "shared" Operationen beliebig viele gleichzeitig Zugriff haben. Dieses Verhalten ist bei ALLEN am Zugriff beteiligten Prozessen korrekt und vollständig einzuprogrammieren (kein "Deadlock", keine "Race conditions" und keine "Starvation"!):

Das sogenannte "Mandatory Locking" (vorgeschrieben, d.h. erzwungen) wird hingegen vom Betriebssystem selber durchgeführt. Es steuert ALLE konkurrierenden Zugriffe auf eine begrenzte Ressource so, dass diese sich nicht gegenseitig in die Quere kommen.

In der Regel ist für Schreiboperationen EIN "EXKLUSIVER" Zugriff auf die Ressource notwendig. Wie der Name schon sagt, sind dann gleichzeitig KEINE weiteren "exklusiven" oder "shared" Zugriffe möglich. Solange KEIN "exklusiver" Zugriff besteht, sind hingegen MEHRERE "SHARED" Zugriffe möglich, die in der Regel eine Leseoperation auf die Ressource darstellen.

```
  +-----+----------+---------+------------+-------+---------+
  |     |  shared  | exklusiv|   shared   | exkl. | shared  |
  +-----+----------+---------+------------+-------+---------+
  | SH1 |ssss...ss.|.........|...ss.......|.......|....ssss.|
  | SH2 |..ssss....|.........|........sss.|.......|.ss......|  .=kein Zugriff
  | SH3 |....sssss.|.........|.sss..ssss..|.......|.....ss..|  s=shared Zugriff
  | EX1 |..........|XXXXXXXXX|............|.......|.........|  X=exklusiver Zg.
  | EX2 |..........|.........|............|XXXXXXX|.........|
  +-----+----------+---------+------------+-------+---------+
           ^          |         ^           |        ^
           X          +------+  X  +--------+        X
           X                 |  X  |                 X
           X                 v  X  v                 X
           X               .-------.                 X
           X               |       |                 X
           #================| DATEI |================#
                           |       |
                           `-------'
```

Bevor ein Prozess also auf die Ressource zugreifen kann, muss er sich erst per "flock" den gerade benötigten Lock-Typ verschaffen. Dabei kann er in der Regel wählen, ob er BLOCKIERT wird, wenn er den Lock nicht erhält oder ob er NICHT BLOCKIERT wird, und dann eben andere Operationen durchführen kann, bevor er wieder versucht, auf die gemeinsame Ressource zuzugreifen.

ACHTUNG: "flock" verhindert nicht die Durchführung konkurrierenden Operationen auf die Ressource "per se" (z.B. read, write), sondern nur, dass gleichzeitig mehr als ein Prozess einen "exklusiven" Lock auf die Ressource setzt.

ACHTUNG: Der Zugriff von anderen Prozessen auf die Ressource wird nicht verhindert, wenn diese nicht SELBST "flock" nutzen, um die Zugriffe auf die Ressource zu konkurrieren. In JEDES Programm muss also der "flock"-Mechanismus eingebaut werden, damit ALLE Programme ihren konkurrierenden Zugriff auf die Ressource sauber koordinieren.

2) Beispiel-Programm

```
#!/usr/bin/perl -w
#-------------------------------------------------------------------------------
# flock.pl
# Beispiel für Filelocking konkurrierender Prozesse.
#-------------------------------------------------------------------------------
use strict;

use Fcntl(":flock");
# Anschließend definierte Konstanten (Funktionen mit konstantem Rückgabewert):
# &LOCK_SH = 1;    # Shared Lock
# &LOCK_EX = 2;    # Exclusive Lock
# &LOCK_NB = 4;    # Nonblocking (nur Zusatz per Bit-Oder "|"!)
# &LOCK_UN = 8;    # Unlock

my $lockfile = "/tmp/lock";
my $anz      = 0;

# Datei für Locking MUSS geöffnet sein
open(FILE, ">>", $lockfile) or die "Kann Datei '$lockfile' nicht öffnen\n";

# Puffern abschalten (nicht unbedingt notwendig)
#$| = 1;

# Endlosschleife
for (;;)
{
    print "$$: (A) Versuche exklusiv zu locken (blocking)\n";
    flock(FILE, &LOCK_EX);
    print "$$: Datei '$lockfile' zum Schreiben gelockt (exclusive)\n";
    print FILE "$$: ", qx(/bin/date), "\n";
    sleep(5);
    flock(FILE, &LOCK_UN);
    print "$$: Lock entfernt (exclusive)\n";

    print "$$: (B) Versuche exklusiv zu locken (nonblocking)\n";
    if (flock(FILE, &LOCK_EX | &LOCK_NB))
    {
        print "$$: Datei '$lockfile' zum Schreiben gelockt (exclusive)\n";
        print FILE "$$: ", qx(/bin/date), "\n";
        sleep(3);
        flock(FILE, &LOCK_UN);
        print "$$: Lock entfernt (exclusive)\n";
    }
    else
    {
        print "$$: Exklusiver Lock gescheitert!!!\n";
    }

    print "$$: (C) Versuche shared zu locken (nonblocking)\n";
    if (flock(FILE, &LOCK_SH | &LOCK_NB))
    {
```

```
        print "$$: Datei '$lockfile' zum Lesen gelockt (shared)\n";

        # Datei einlesen und Anzahl Zeilen zählen (2. Dateihandle notwendig!)
        open(INPUT, "<", $lockfile) or
            die "Kann Datei '$lockfile' nicht zum lesen öffnen";
        my @zeilen = <INPUT>;
        $anz = scalar @zeilen;
        close(INPUT);
        print "$$: $anz Zeilen darin gefunden\n";
        sleep(1);
        flock(FILE, &LOCK_UN);
        print "$$: Lock entfernt (shared)\n";
    }
    else
    {
        print "$$: Shared Lock gescheitert!!!\n";
    }

    sleep(2);
}

close(FILE);
```