LoxBerry::System::lock

$check = LoxBerry::System::lock([lockfile => 'lockname'], [wait => $seconds]);

Erstellt ein Lock-File, das die PID des laufenden Prozesses enthält. Dieses Lock-File kann nur erstellt werden, wenn keine wichtigen Systemprozesse laufen. Es ist möglich, eine angegebene Zeit abzuwarten, sollte der Lock nicht freigegeben sein.Übergeben wird ein `lockname` ohne Pfad und Dateierweiterung.

Die Funktion geht bei der Ausführung folgendermaßen vor:

1. Es wird die Datei `$lbsconfigdir/lockfiles.default` ausgelesen, die alle Namen der Lockfiles enthält, die zum Zeitpunkt dieses Locks nicht laufen dürfen.
2. Für alle diese Lockfiles wird geprüft, ob deren PID noch existiert.
3. Existiert diese PID noch, wird entweder im Intervall 5 Sekunden die optionale Zeit `wait => <Sekunden>` gewartet, oder als String der gelockte Prozess zurückgegeben.
4. Existiert die PID nicht mehr, wird versucht, das entsprechende Lock-File zu entfernen, wenn ausreichend Dateirechte vorhanden sind.
5. Nachdem die Liste aller Lockfiles abgearbeitet wird, wird der optionale `lockfile => 'lockname'` erzeugt.
6. Auch hier wird, abhängig von `wait`, entweder wiederholt versucht zu locken, oder im Fehlerfall sofort der Lockname als String zurückgegeben.

Zusammenfassend

- Wird `lockfile` angegeben, wird versucht, einen Lock für dieses File zu erstellen. Ohne Angabe werden nur die `lockfiles.default` Dateien überprüft.
- Wird `wait` *nicht* angegeben, wird beim ersten Auftreten eines Locks der sperrende Lockname als String zurückgegeben.
- Wird `wait` angegeben, wird für *jede* Datei die angegebene Zeit gewartet. Die Zeit vervielfacht sich deswegen, wenn mehrere Dateien gerade gesperrt sind (was grundsätzlich gar nicht oder nur durch "Pech" möglich sein dürfte).
- Die Funktion liefert immer **undef** zurück, wenn der Lock **erfolgreich** war.
- Die Funktion liefert immer den **String** des sperrenden Lockfiles zurück, wenn der Lock **nicht erfolgreich** war.
- Beim Erstellen der Datei werden die Dateirechte auf 0666 gesetzt.
- Kann die Funktion ein veraltetes Lock-File, dessen Prozess nicht mehr existiert, dennoch nicht löschen, wird dies als **nicht erfolgreich** zurückgegeben (es kommt dann die Fehlermeldung als String zurück).

Zum Entsperren wird LoxBerry::System::unlock(lockfile => 'lockname'); verwendet, siehe LoxBerry::System::unlock.

Debugging

Generell können in den LoxBerry Perl-Modulen Debugging-Meldungen für alle Funktionen aktiviert werden, die dann ins Errorlog (STDERR) geschrieben werden:

```
$LoxBerry::System::DEBUG = 1;
```

Für diese Funktion gibt es sehr detaillierte Debugging-Meldungen.

Verwendung

```
use LoxBerry::System;

# Locks lbupdate and immediately returns, if it cannot lock
my $status = LoxBerry::System::lock(lockfile => 'lbupdate');
if ($status) {
        print "Could not lock, prevented by $status";
} else {
        print "Lock was successful";
}


# Locks myprocedure and will retry it for 10 minutes, if it cannot lock
my $status = LoxBerry::System::lock(lockfile => 'myprocedure', wait => 600);


# Only checks, if important files are locked, and immediately returns
my $status = LoxBerry::System::lock();
if ($status) {
        print "$status currently running - Quitting.";
        exit;
}
```

Known Issues und Informationen

- Die Verwendung der PID könnte jetzt oder in Zukunft Schrierigkeiten bereiten:
  - In einer FastCGI oder mod_perl Konfiguration ist unklar, ob ein Lock eine eigene PID erhält, oder die Webserver-PID verwendet wird.
  - Ein vergessenes unlock - wenn die Webserver-PID herangezogen wird, könnte dazu führen, dass diese PID nicht mehr gelockt werden kann.
- Soweit die Rechte ausreichen, kann problemlos ein unlock ohne vorherigem Lock eines fremden Prozesses durchgeführt werden. Derzeit prüft unlock nicht die im Lockfile enthaltene PID.
- Der Pfad der Lockfiles ist /var/lock/

Siehe auch

LoxBerry::System::unlock