

```

Sep 19 14:20:18 amd64 sshd[20494]: Accepted rsa for esser from ::ffff:87.234.201.207 port 61557
Sep 19 14:20:41 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 01:00:01 amd64 /usr/sbin/cron[29278]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 20 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 02:00:01 amd64 /usr/sbin/cron[30103]: (root) CMD (/sbin/evlogmgr -c "age > *30d")
Sep 20 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:46:44 amd64 sshd[6516]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62004
Sep 20 12:46:44 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:48:41 amd64 sshd[6609]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62105
Sep 20 12:54:44 amd64 sshd[6694]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62514
Sep 20 15:27:35 amd64 sshd[9077]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64242
Sep 20 16:37:11 amd64 sshd[10102]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63375
Sep 20 16:37:11 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 16:38:10 amd64 sshd[10140]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63546
Sep 21 01:00:01 amd64 /usr/sbin/cron[17055]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 21 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 02:00:01 amd64 /usr/sbin/cron[17878]: (root) CMD (/sbin/evlogmgr -c "age > *30d")
Sep 21 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 17:43:26 amd64 sshd[31088]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63397
Sep 21 17:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 17:53:39 amd64 sshd[31269]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64391
Sep 21 18:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 19:43:26 amd64 sshd[31269]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64391
Sep 22 01:00:01 amd64 /usr/sbin/cron[24739]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 22 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 02:00:01 amd64 /usr/sbin/cron[24739]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 22 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 02:00:01 amd64 /usr/sbin/cron[24739]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 22 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 20:23:21 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 01:00:01 amd64 /usr/sbin/cron[24739]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 23 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 02:00:01 amd64 /usr/sbin/cron[24739]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 23 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 02:00:01 amd64 /usr/sbin/cron[24739]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 23 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:05 amd64 sshd[6554]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62093
Sep 23 18:04:05 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:34 amd64 sshd[6606]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62093
Sep 24 01:00:01 amd64 /usr/sbin/cron[12436]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 24 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 02:00:01 amd64 /usr/sbin/cron[13253]: (root) CMD (/sbin/evlogmgr -c "age > *30d")
Sep 24 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 11:15:48 amd64 sshd[21317]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64456
Sep 24 11:15:48 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 13:49:08 amd64 sshd[21317]: Accepted rsa for esser from ::ffff:87.234.201.207 port 61330
Sep 24 13:49:08 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 15:42:07 amd64 kernel: snd_seq_midi_event: unsupported module, tainting kernel.
Sep 24 15:42:07 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 15:42:07 amd64 kernel: snd_seq_oss: unsupported module, tainting kernel.
Sep 24 20:25:31 amd64 sshd[29399]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62566
Sep 24 20:25:31 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 01:00:02 amd64 /usr/sbin/cron[6621]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 25 01:00:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 02:00:01 amd64 /usr/sbin/cron[1484]: (root) CMD (/sbin/evlogmgr -c "age > *30d")
Sep 25 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 02:00:02 amd64 /usr/sbin/cron[1484]: (root) CMD (/sbin/evlogmgr -c "age > *30d")
Sep 25 02:00:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:25 amd64 sshd[8889]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64183
Sep 25 10:59:25 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:47 amd64 sshd[8921]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64253
Sep 25 11:30:02 amd64 sshd[9372]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62029
Sep 25 11:59:25 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:05:37 amd64 sshd[11554]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62822
Sep 25 14:05:37 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:06:10 amd64 sshd[11586]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62951
Sep 25 14:07:12 amd64 sshd[11608]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63392
Sep 25 14:08:33 amd64 sshd[11630]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63709
Sep 25 15:25:33 amd64 sshd[12930]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62778

```

6. Inter-Prozess-Kommunikation (1)

6. IPC 6.1 Einführung

/home/esser/Daten/Dozent/Folien/bs-esser-17.cdp

Synchronisation mit Python (1)

- 100 Threads manipulieren eine Variable
- Ergebnis der Berechnung sollte 0 sein
- Problem: kritischer Bereich beim Zugriff auf globale Variable

```

$ ./test.py
Ergebnis: 200
$ ./test.py
Ergebnis: 800
$ ./test.py
Ergebnis: 0
$ ./test.py
Ergebnis: 100
$ ./test.py
Ergebnis: 0

```

```

from threading import Thread

class testthread(Thread):
    def __init__(self):
        Thread.__init__(self)
    def run(self):
        global globalcount
        # Anfang kritischer Bereich
        for j in range(0,99999):
            globalcount += 100
            globalcount -= 100
        # Ende kritischer Bereich

globalcount=0 # glob. Variable
threads = []
for i in range(0,100): # Threads starten
    t = testthread()
    threads.append(t)
    t.start()
for t in threads: t.join() # aufräumen

print "Ergebnis:",globalcount

```

```

Sep 19 14:27:41 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 01:00:01 amd64 /usr/sbin/cron[29278]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 20 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 02:00:01 amd64 /usr/sbin/cron[30103]: (root) CMD (/sbin/evlogmgr -c "age > *30d")
Sep 20 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:46:44 amd64 sshd[6516]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62004
Sep 20 12:46:44 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:48:41 amd64 sshd[6609]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62105
Sep 20 12:54:44 amd64 sshd[6694]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62514
Sep 20 15:27:35 amd64 sshd[9077]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64242
Sep 20 16:37:11 amd64 sshd[10102]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63375
Sep 20 16:37:11 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 16:38:10 amd64 sshd[10140]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63546
Sep 21 01:00:01 amd64 /usr/sbin/cron[17055]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 21 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 02:00:01 amd64 /usr/sbin/cron[17878]: (root) CMD (/sbin/evlogmgr -c "age > *30d")
Sep 21 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 17:43:26 amd64 sshd[31088]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63397
Sep 21 17:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 17:53:39 amd64 sshd[31269]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64391
Sep 21 18:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 19:43:26 amd64 sshd[31269]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64391
Sep 22 01:00:01 amd64 /usr/sbin/cron[24739]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 22 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 02:00:01 amd64 /usr/sbin/cron[24739]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 22 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 20:23:21 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 01:00:01 amd64 /usr/sbin/cron[24739]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 23 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 02:00:01 amd64 /usr/sbin/cron[24739]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 23 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:05 amd64 sshd[6554]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62093
Sep 23 18:04:05 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:34 amd64 sshd[6606]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62093
Sep 24 01:00:01 amd64 /usr/sbin/cron[12436]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 24 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 02:00:01 amd64 /usr/sbin/cron[13253]: (root) CMD (/sbin/evlogmgr -c "age > *30d")
Sep 24 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 11:15:48 amd64 sshd[21317]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64456
Sep 24 11:15:48 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 13:49:08 amd64 sshd[21317]: Accepted rsa for esser from ::ffff:87.234.201.207 port 61330
Sep 24 13:49:08 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 15:42:07 amd64 kernel: snd_seq_midi_event: unsupported module, tainting kernel.
Sep 24 15:42:07 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 15:42:07 amd64 kernel: snd_seq_oss: unsupported module, tainting kernel.
Sep 24 20:25:31 amd64 sshd[29399]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62566
Sep 24 20:25:31 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 01:00:02 amd64 /usr/sbin/cron[6621]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 25 01:00:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 02:00:01 amd64 /usr/sbin/cron[1484]: (root) CMD (/sbin/evlogmgr -c "age > *30d")
Sep 25 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 02:00:02 amd64 /usr/sbin/cron[1484]: (root) CMD (/sbin/evlogmgr -c "age > *30d")
Sep 25 02:00:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:25 amd64 sshd[8889]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64183
Sep 25 10:59:25 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:47 amd64 sshd[8921]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64253
Sep 25 11:30:02 amd64 sshd[9372]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62029
Sep 25 11:59:25 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:05:37 amd64 sshd[11554]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62822
Sep 25 14:05:37 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:06:10 amd64 sshd[11586]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62951
Sep 25 14:07:12 amd64 sshd[11608]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63392
Sep 25 14:08:33 amd64 sshd[11630]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63709
Sep 25 15:25:33 amd64 sshd[12930]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62778

```

Nachtrag zu Kap. 5: Synchronisation mit Python (→ Praktikum)

Synchronisation mit Python (2)

1. Lösung: Lock

- acquire: sperren
- release: freigeben

```

$ ./lock.py
Ergebnis: 0
$ ./lock.py
Ergebnis: 0
$ ./lock.py
Ergebnis: 0
$ ./lock.py
Ergebnis: 0
$ time ./test.py
real 0m5.560s
$ time ./lock.py
real 0m5.560s

```

(kein messbarer Unterschied)

```

from threading import Thread, Lock

class testthread(Thread):
    def __init__(self):
        Thread.__init__(self)
    def run(self):
        global globalcount
        # Anfang kritischer Bereich
        mylock.acquire()
        for j in range(0,99999):
            globalcount += 100
            globalcount -= 100
        mylock.release()
        # Ende kritischer Bereich

globalcount=0 # glob. Variable
threads = []
mylock = Lock() # globales Lock
for i in range(0,100): # Threads starten
    t = testthread()
    threads.append(t)
    t.start()
for t in threads: t.join() # aufräumen

print "Ergebnis:",globalcount

```

Synchronisation mit Python (3)

- **Lock**
 - acquire kann auch nicht-blockierend aufgerufen werden: **mylock.acquire(0)**
 - wenn das Lock verfügbar ist, gibt **acquire()** **True** zurück (und gibt dem aufrufenden Thread das Lock)
 - wenn schon ein anderer Thread das Lock hält, wartet **acquire()** nicht, sondern gibt sofort **False** zurück.
- **RLock (Reentrant Lock)**
 - wie Lock, aber: kann rekursiv benutzt werden
 - Lock so oft mit **release()** freigeben, wie es mit **acquire()** erworben wurde

Synchronisation mit Python (5)

- **BoundedSemaphore**
 - für Verwaltung mehrerer Ressourcen: mit größerem Zählerwert starten
 - **acquire** = **Wait-Operation**
 - mit Argument 0: nicht-blockierend
 - **release** = **Signal-Operation**

Synchronisation mit Python (4)

2. Lösung: BoundedSemaphore

- acquire: runter zählen
- release: hoch zählen

```
$ ./lock.py
Ergebnis: 0
$ ./lock.py
Ergebnis: 0
$ ./lock.py
Ergebnis: 0
$ ./lock.py
Ergebnis: 0

$ time ./test.py
real    0m5.560s
$ time ./lock.py
real    0m5.560s
```

```
from threading import Thread, BoundedSemaphore

class testthread(Thread):
    def __init__(self):
        Thread.__init__(self)
    def run(self):
        global globalcount
        # Anfang kritischer Bereich
        mysem.acquire()
        for j in range(0,99999):
            globalcount += 100
            globalcount -= 100
        mysem.release()
        # Ende kritischer Bereich

globalcount=0 # glob. Variable
threads = []
mysem = BoundedSemaphore(1) # init: 1
for i in range(0,100): # Threads starten
    t = testthread()
    threads.append(t)
    t.start()
for t in threads: t.join() # aufräumen

print "Ergebnis:",globalcount
```

Synchronisation mit Python (6)

- **Condition: Bedingungsvariablen**
 - vgl. Java / Monitor
 - Bedingungsvariable ist immer mit einem Lock verbunden
 - Funktionen
 - cv.acquire ()** zugehöriges Lock erhalten (oder blockieren)
 - cv.release ()** zugeh. Lock freigeben
 - cv.wait ()** Lock freigeben und blockieren, bis Signal kommt
 - cv.notify ()** einen (auf **cv**) wartenden Thread wecken
 - cv.notifyAll ()** alle (auf **cv**) wartenden Threads wecken

Synchronisation mit Python (7)

Condition: Producer-Consumer-Problem

```
from threading import Thread, Condition
cv = Condition()

# Consume one item
cv.acquire()
while not an_item_is_available():
    cv.wait()
get_an_available_item()
cv.release()

# Produce one item
cv.acquire()
make_an_item_available()
cv.notify()
cv.release()
```

```
Sep 19 14:20:18 amd64 sshd[20494]: Accepted rsa for esser from ::ffff:87.234.201.207 port 61557
Sep 19 14:27:43 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 01:00:01 amd64 /usr/sbin/cron[29278]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 20 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 02:00:01 amd64 /usr/sbin/cron[31031]: (root) CMD (/sbin/evlogmgr -c 'age > *30d*')
Sep 20 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:46:44 amd64 sshd[6516]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62004
Sep 20 12:46:44 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:48:41 amd64 sshd[6609]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62105
Sep 20 12:54:44 amd64 sshd[6694]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62514
Sep 20 15:27:35 amd64 sshd[9077]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64242
Sep 20 15:27:35 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 16:37:11 amd64 sshd[10102]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63375
Sep 20 16:37:11 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 16:38:10 amd64 sshd[10440]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63546
Sep 21 01:00:01 amd64 /usr/sbin/cron[17055]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 21 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 02:00:01 amd64 /usr/sbin/cron[17978]: (root) CMD (/sbin/evlogmgr -c 'age > *30d*')
Sep 21 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 17:43:26 amd64 sshd[31086]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63397
Sep 21 17:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 17:53:39 amd64 sshd[31269]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64391
Sep 21 18:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 19:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 01:00:01 amd64 /usr/sbin/cron[4674]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 22 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 02:00:01 amd64 /usr/sbin/cron[5499]: (root) CMD (/sbin/evlogmgr -c 'age > *30d*')
Sep 22 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 20:23:21 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 01:00:01 amd64 /usr/sbin/cron[24391]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 23 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 02:00:01 amd64 /usr/sbin/cron[25555]: (root) CMD (/sbin/evlogmgr -c 'age > *30d*')
Sep 23 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:05 amd64 sshd[6541]: Accepted publickey for esser from ::ffff:87.234.201.207 port 61919
Sep 23 18:04:05 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:34 amd64 sshd[6606]: Accepted rsa for esser from ::ffff:87.234.201.207 port 61919
Sep 24 01:00:01 amd64 /usr/sbin/cron[12486]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 24 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 02:00:01 amd64 /usr/sbin/cron[13253]: (root) CMD (/sbin/evlogmgr -c 'age > *30d*')
Sep 24 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 11:15:48 amd64 sshd[20998]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64456
Sep 24 11:15:48 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 13:49:08 amd64 sshd[21397]: Accepted rsa for esser from ::ffff:87.234.201.207 port 61330
Sep 24 13:49:08 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 15:42:07 amd64 kernel: snd_seq_midi_event: unsupported module, tainting kernel.
Sep 24 15:42:07 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 15:42:07 amd64 kernel: snd_seq_oss: unsupported module, tainting kernel.
Sep 24 20:25:31 amd64 sshd[29399]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62566
Sep 24 20:25:31 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 01:00:02 amd64 /usr/sbin/cron[6621]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 25 01:00:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 02:00:01 amd64 /usr/sbin/cron[1484]: (root) CMD (/sbin/evlogmgr -c 'age > *30d*')
Sep 25 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:25 amd64 sshd[8889]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64183
Sep 25 10:59:25 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:47 amd64 sshd[8921]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64253
Sep 25 11:30:02 amd64 sshd[9372]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62029
Sep 25 11:59:25 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:05:37 amd64 sshd[11554]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62822
Sep 25 14:05:37 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:06:10 amd64 sshd[11586]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62951
Sep 25 14:07:13 amd64 sshd[11606]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63392
Sep 25 14:08:33 amd64 sshd[11620]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63709
Sep 25 15:25:33 amd64 sshd[12930]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62778
```

6.1 IPC: Einführung

Synchronisation mit Python (8)

Event: entspricht „manuellen Events“ (Windows)

```
from threading import Thread, Event
ev = Event()

ev.set()                    # Event setzen
ev.clear()                # Event zurücksetzen
ev.isSet()                # Status abfragen
ev.wait()                 # Blockieren, bis Status auf
                          gesetzt wechselt
```

Ausführlichere Darstellung in der Python-Dokumentation:
<http://docs.python.org/lib/module-threading.html>

Inter Process Communication

- Nachrichtenaustausch zwischen mehreren Prozessen oder Threads
- Verbindung durch Kommunikationssystem
- Sender und Empfänger
- Nötig, wenn es keinen gemeinsamen Hauptspeicher gibt
- eine weitere Synchronisationsform

IPC-Charakterisierung (1)

- Kommunikationsmodell:
 - Punkt-zu-Punkt-Kommunikation
 - Publish-Subscribe-Kommunikation
 - Broadcast-Kommunikation
- Übertragungsrichtung:
 - simplex / unidirektional
 - duplex / bidirektional
- Synchronität
 - synchron / blockierend
 - asynchron / nicht-blockierend

Kommunikationsmodell

- Punkt zu Punkt
 - genau ein Sender und ein Empfänger
- Broadcast
 - ein Sender und mehrere Empfänger
- Publish & Subscribe
 - Peer-to-peer-Kommunikation
 - Publisher: Threads, die eine Eigenschaft ändern
 - Subscriber: Threads, die bei solchen Änderungen benachrichtigt werden

IPC-Charakterisierung (2)

- Art der Nachricht
 - Nachrichten- oder
 - Stream-orientiert
- Plattform(-un-)abhängigkeit
- Portierbarkeit
- Lokalität
 - systemgebunden oder
 - Netzwerkkommunikation möglich (über Rechengrenzen hinweg)

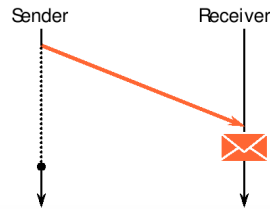
Übertragungsrichtung

- Simplex-Nachricht vom Sender zum Empfänger
 - Beginn: Absenden beim Sender, Ende: Zustellung beim Empfänger
 - Keine Antwort erwartet
- Duplex-Nachricht
 - Beginn: Absenden des Auftrags beim Sender, Ende: Zustellung der Erfolgsbestätigung (Quittung) beim Sender
 - Dazwischen: Auftragsbearbeitung auf Empfängerseite
 - Im einfachsten Fall zwei Nachrichten
 - Ausbleibende Quittungen durch Timeouts erkennen (negative Quittung)

Synchronität

Synchrone / blockierende Kommunikation

- Sender blockiert, bis Nachricht ankommt
- Benötigt fast keine Pufferkapazität
- beschränkte Parallelität



Asynchrone / nicht block. Kommunikation

- Sender blockiert nur, bis Nachricht in Puffer kopiert wurde
- benötigt größere Pufferkapazität
- Sender kann mehrere Nachrichten kurz nacheinander abschicken

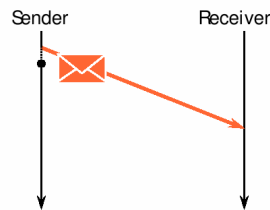
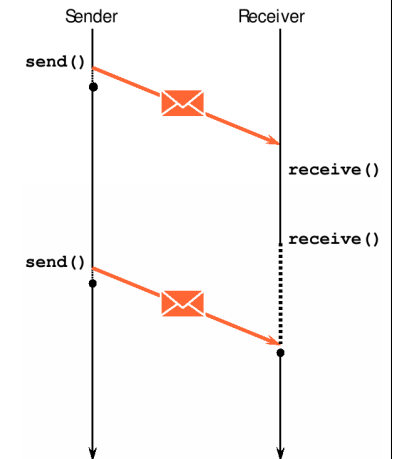


Bild: (c)Peter Sturm, Uni Trier, „Episodes on Operating Systems“
http://strathisla.uni-trier.de/lectblog/wp-content/Syssoft1_13_Communication.pdf

-1- asynchrone Nachricht

- Sender und Empfänger entkoppelt -> Parallelität
- **UDP**: User Datagram Protocol (IP-basiert)
- Signale: Linux Software Interrupts



Auch die Bilder auf den folgenden vier Folien:
 (c) Peter Sturm, Uni Trier, „Episodes on Operating Systems“,
http://strathisla.uni-trier.de/lectblog/wp-content/Syssoft1_13_Communication.pdf

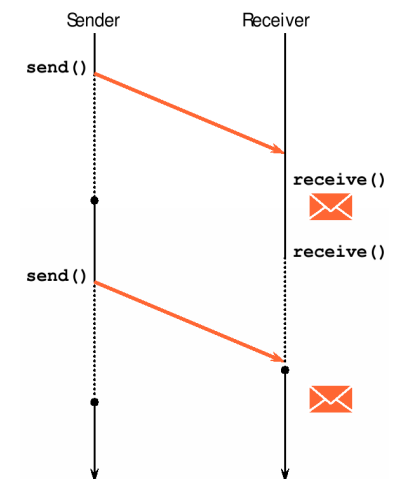
Vier elementare Kommunikationsarten

Aus zwei Übertragungsrichtungen und der Wahl synchron/asynchron ergeben sich:

	asynchron	synchron
simplex: Nachricht	-1- asynchrone Nachricht	-2- synchrone Nachricht
duplex: Auftrag	-3- asynchroner Auftrag	-4- synchroner Auftrag

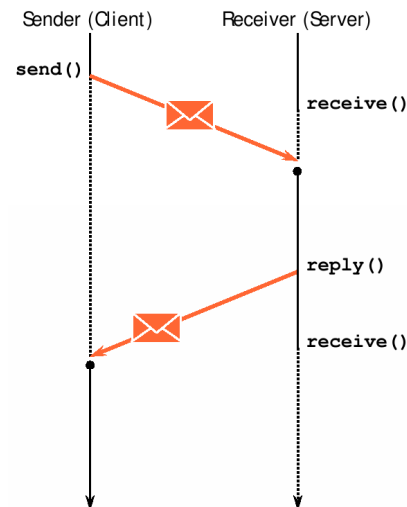
-2- synchrone Nachricht

- Eingeschränkte Parallelität: Sender muss auf Bestätigung durch Empfänger warten
- keine Puffer nötig
- Sender weiß, dass Nachricht angekommen ist



-4- synchroner Auftrag

- **RPC: Remote Procedure Call**
- Sender wartet, bis er auf seine Anfrage die Antwort [nicht nur die Bestätigung] erhalten hat
- z. B. Datenbanken

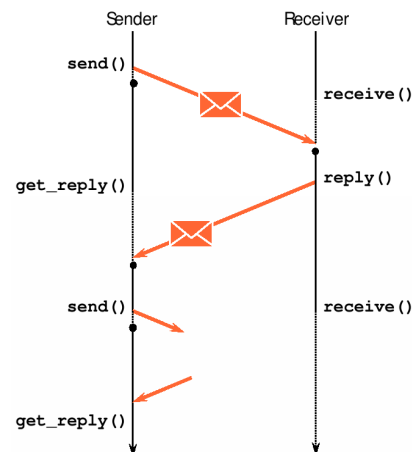


Sockets (1)

- Sockets sind eine Netzwerkschnittstelle, die einen Kommunikationskanal mit folgenden Eigenschaften bereitstellt:
 - sie erlauben IPC zwischen Prozessen
 - lokal oder rechnerübergreifend
 - über verschiedene Protokolle
 - plattformübergreifend
- bidirektionale Kommunikation

-3- asynchroner Auftrag

- Client schickt Anfrage, wartet aber nicht auf das Ergebnis
- Erst zu späterem Zeitpunkt holt er gezielt die Antwort des Servers ab
- Darum: weniger Blockierzeiten, mehr Parallelität



Sockets (2)

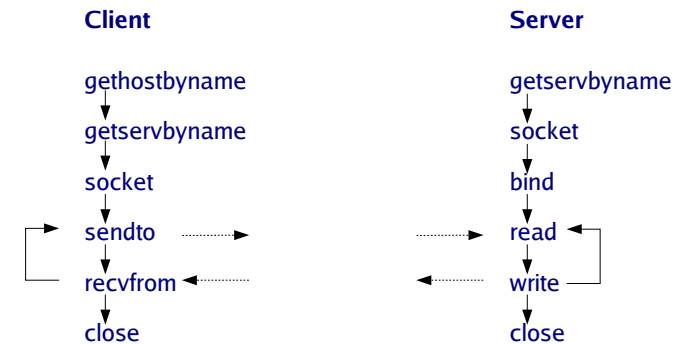
- Adressierung, z.B.
 - lokal (AF_UNIX): Pfadname
 - Netzwerkadresse (AF_INET): host + port, (lokal über AF_INET: host = localhost, 127.0.0.1)
- Zuverlässigkeit:
 - reliable, verbindungsorientiert (z.B. **TCP**-Protokoll):
 - fehlerfrei, keine Verluste, keine Duplikate, Reihenfolge korrekt
 - unreliable, verbindungslos (z.B. **UDP**-Protokoll / Datagram)

Sockets (3)

- **Kommunikationsarten:**
 - Nachrichten-orientiert: `recvmsg()` oder `sendmsg()`
 - Stream-orientiert: `sendto()`, `recvfrom()`, `read()`, `write()`
 - üblicherweise mit 8 kByte gepuffert
 - Senden blockiert bei umfangreicheren Daten, bis Gegenstelle gelesen hat.
 - optional nicht-blockierendes Verhalten wählbar
 - werden in Unix / Linux auf Dateideskriptoren abgebildet (Zugriff ähnlich wie auf Dateien)

Verbindungslose Sockets

Verbindungslose Kommunikation über Datagramme / UDP

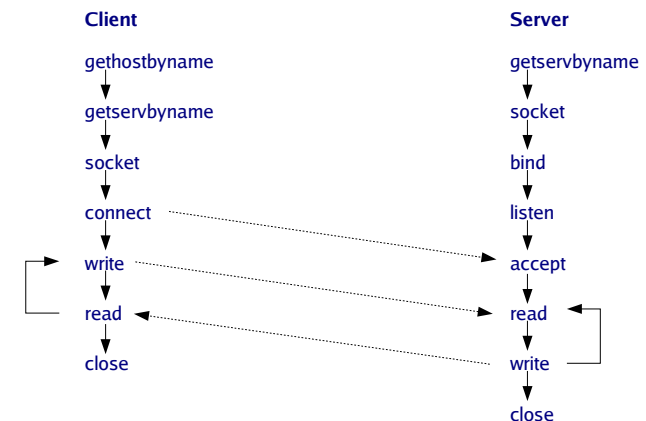


Sockets (4)

- **Anwendung**
 - Mögliche Probleme:
 - Port belegt --> `lsof` ('list open files'-Kommando)
 - Port noch belegt --> ggf. `linger`-Option setzen
 - Nachrichtengrenzen bleiben ggf. nicht erhalten:
 - z.B. `send(170 Byte) + send(230 Byte) = receive(400 Byte)`
 - API-Funktionen erlauben nichtblockierendes Verhalten optional zu wählen.
 - Für eigene Experimente: C-Beispiele für Client-Server am Ende

Verbindungsorientierte Sockets

Verbindungsorientierte Kommunikation über Streams / TCP



Datagram-Server

```
#include <unistd.h> // read(), close()
#include <arpa/inet.h> // sockaddr_in, INADDR_ANY
#include <sys/socket.h> // SOCK_DGRAM, socket(), bind()

const short port = 5242;
int n, sockfd;
char buf[256];
struct sockaddr_in serv_addr;

int main() { // UDP_Socket
    if ((sockfd = socket( AF_INET, SOCK_DGRAM, 0 )) < 0) perror("opening datagram");

    // Create name with wildcards
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons( port );

    if ( bind( sockfd, (sockaddr *)&serv_addr, sizeof(serv_addr) ) != 0 )
        perror( "binding to address" );

    n = read( sockfd, buf, sizeof(buf) ); printf( "Received: %s\n", buf );

    close( sockfd ); return 0;
}
```

Server verbindungsorientiert

```
#include <unistd.h> // read(), close()
#include <arpa/inet.h> // sockaddr_in, INADDR_ANY
#include <sys/socket.h> // SOCK_DGRAM, socket(), bind()

const short port = 5242, waitqueuelen = 1;
int n, sockfd, con;
char buf[256];
struct sockaddr_in serv_addr;

int main() { // TCP_Socket
    if ((sockfd = socket( AF_INET, SOCK_STREAM, 0 )) < 0) perror("opening stream");
    serv_addr...;
    if ( bind( sockfd, (sockaddr *)&serv_addr, sizeof(serv_addr) ) != 0 )
        perror( "binding to address" );

    if ( listen( sockfd, waitqueuelen ) != 0 ) perror( "listening to address" );

    if ( ( con = accept( sockfd, (sockaddr *)&peer_addr, sizeof(serv_addr) ) < 0 )
        perror( "accepting client" );

    n = read(con, buf, sizeof(buf)); printf("Received: %s\n", buf); write(con, buf, n);

    close( con ); close( sockfd ); return 0;
}
```

Datagram-Client

```
#include <unistd.h> // read(), close()
#include <arpa/inet.h> // sockaddr_in, AF_INET
#include <sys/socket.h> // SOCK_DGRAM, socket(), bind()
#include <sys/param.h> // MAXHOSTNAMELEN
#include <netdb.h> // gethostbyname()

const short port = 5242;
char hostname[MAXHOSTNAMELEN+1] = "server";
int sockfd;
struct sockaddr_in peer_addr;

int main() { // UDP_Socket
    if ((sockfd = socket( AF_INET, SOCK_DGRAM, 0 )) < 0) perror("opening datagram");

    struct hostent * hp = gethostbyname( hostname );
    bcopy( hp->h_addr, (char *)&peer_addr.sin_addr, hp->h_length );
    peer_addr.sin_family = AF_INET;
    peer_addr.sin_port = htons( port );

    if ( sendto( sockfd, "Hello World", 11, 0, (sockaddr *)&peer_addr,
        sizeof(peer_addr) ) < 0 ) perror( "sending data" );

    close( sockfd ); return 0;
}
```

Client verbindungsorientiert

```
#include <unistd.h> // read(), close()
#include <arpa/inet.h> // sockaddr_in, AF_INET
#include <sys/socket.h> // SOCK_DGRAM, socket(), bind()
#include <sys/param.h> // MAXHOSTNAMELEN
#include <netdb.h> // gethostbyname()

const short port = 5242;
char hostname[MAXHOSTNAMELEN+1] = "server";
int sockfd;
struct sockaddr_in peer_addr;

int main() { // TCP_Socket
    if ((sockfd = socket( AF_INET, SOCK_STREAM, 0 )) < 0) perror("opening stream");
    peer_addr...;
    if ( connect( sockfd, (sockaddr *)&peer_addr, sizeof(peer) ) != 0 )
        perror( "connecting server" );

    write( sockfd, "Hello World", 11 );
    read( sockfd, buf, sizeof(buf)); printf("Received: %s\n", buf);

    close( sockfd ); return 0;
}
```