

BS Zusatz - GÜ

Klausur: bs1\_2008h.pdf (L2P)

a1.)~~Ressourcen~~

P: Semaphore sperren (runterzählen)

blockieren bis S. verfügbar

V: Semaphore freigeben  
(rautzählen)a2.)

int test\_and\_set(int \*v)

{ int tmp = \*v;

\*v = 1;

return tmp;

}

void P(int \*s)

{ while (test\_and\_set(s));

}

void V(int \*s)

{ \*s = 0;

}

a3.)

P(S);

C++

V(S);

← zählt semaphore mittels  
binärem Sem.

b.2.) id=0 ist nötig, da sonst ~~Prozesse~~ <sup>Threads</sup> mit niedriger ID vorkommen könnten

b.2.1) Nein, Threads mit niedrigen IDs werden bevorzugt.

b.3.) Der Code implementiert busy-waiting.

c1.)

```
....  
unsigned int reader-counter;  
semaphor    reader-counter-lock;  
semaphor    writer-lock;  
....
```

```
void get-coordinates (elem_t * elem,  
                      double * x, double * y)
```

```
{  
    { P(&elem->reader-counter-lock);  
      (elem->reader-counter)++;  
      V(&(elem->reader-counter-lock));  
      *x = elem->x;  
      *y = elem->y;  
      P(...)  
      (elem->...)--;  
      V(...)  
    }  
}
```

ÜS Zusatz - Gü

```
void set_coordinates(elem_t * elem,
                     double x, double y)
```

```
{
    P(&(elem->writer_lock));
    P(&(elem->reader_counter_lock));
    while (elem->reader_counter > 0) {
        V(&(elem->reader_counter_lock));
        // nap; nap;
        P(&(elem->reader_counter_lock));
    }
    elem->x = ...
    elem->y = ...
    V(&(elem->reader_counter_lock));
    V(&(elem->writer_lock));
}
```

