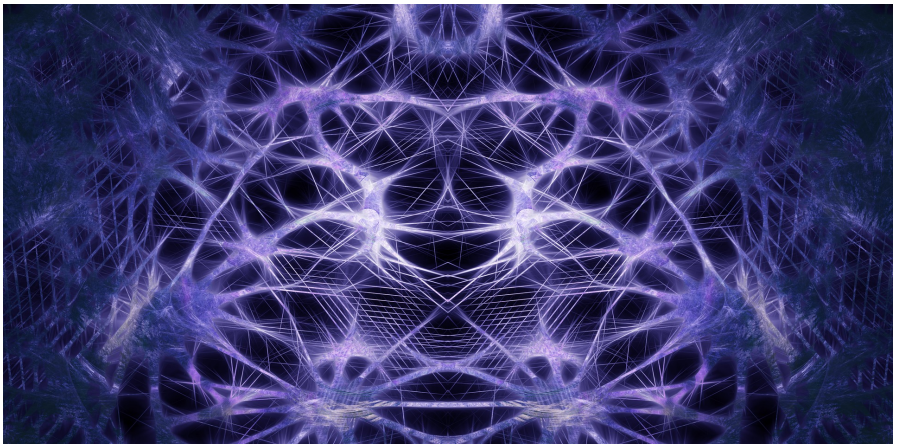

Übung 04

Nichtsequentielle und Verteilte Programmierung

Sebastian Faase, Lutz Schäfer

Tutorium Leon Dirmeier

Freie Universität Berlin

16. Mai 2019

I Speisende Philisophen in C (12 Punkte)

Um unsere Implementierung mit den Vorgaben dieser Aufgabe zu starten, nutzten wir

```
./sim -n 5 -a 10
```

alternativ auch

```
./sim
```

da die Parameter als *default* gewählt wurden.

Die Simulation hat einige Kommandozeilenparameter, die wir in der Simulation des Tisches behandeln [I.1]. Der Tisch an dem die Philosophen sitzen wird wie folgt gestartet [I.2]. Er wird mit Chopsticks [I.3] und Philosophen initialisiert [I.4]. Die Philosophen können leben [I.5], essen [I.6] und denken [I.7].

Da unsere Ausgabe hundert Zeilen pro Sekunde erzeugt, sei hier nur ein Auszug der Ausgabe Wiedergegeben [I.8].

Listing I.1: simulation.c

```
int main(int argc, char *argv[])
{
  int64_t help_is_set = 0;
  Parameter parameter = {.n = 5,
                         .age = 10 /* timeout  */};

  /* parsing commandline arguments */
  int64_t opt;
  while ( -1 != (opt = getopt(argc, argv, "hn:a:")) )
  {
    switch (opt) {
      case 'h':
        printhelp(stdout, argv[0]);
        help_is_set = 1;
        break;
      case 'n':
        parameter.n = atoi(optarg); // TODO use strtol
        /* if there is no philos or we are in an alternete universe abort */
        if ( 3 > parameter.n ) {
          printhelp(stderr, argv[0]);
          exit(EXIT_FAILURE);
        }
        break;
      case 'a': /* this serves as a timeout */
        parameter.age = atoi(optarg);
        if (1 > parameter.age) {
          printhelp(stderr, argv[0]);
          exit(EXIT_FAILURE);
        }
        break;
      default: /* .?. */
        printhelp(stderr, argv[0]);
```

```
        exit(EXIT_FAILURE);
      }
    }


    /* if no help shall be displayed, start the world */
    if (!help_is_set)
    {
      run(parameter);
    }

    exit(EXIT_SUCCESS);
}
```

Listing I.2: table.c⟶run

```
void run(Parameter parameter)
{

    size_t number_of_philosophs = parameter.n;
    ALL = number_of_philosophs; /* used for making them all simultaniously */
    size_t age = parameter.age;

    /* create data structures in order */
    create_sticks(number_of_philosophs);
    create_philosoph(number_of_philosophs, age);
    for (size_t i = 0; i < number_of_philosophs; i++) {
      if ( 0 != (pthread_create(&philosoph_l[i].thread,
                                NULL,
                                (void *)live,
                                 &philosoph_l[i])))
      {
        /* abort if not all threads can be created */
        perror("cannot create pthread");
        exit(EXIT_FAILURE);
      }
    }
    /* Broadcast solution to start "all" threads simultaniously */
    pthread_t broadcaster;
    sem_init(&THERE, 0, number_of_philosophs);
    printf("Number of Philosophs: %ld\n",number_of_philosophs);
    int i=-1;
    sem_getvalue(&THERE, &i);
    pthread_create(&broadcaster,NULL, (void *) broadcast, &cond);

    for (size_t i = 0; i < number_of_philosophs; i++) {
      pthread_join(philosoph_l[i].thread, NULL);
    }

    pthread_cancel(broadcaster);
```

```
  exit(EXIT_SUCCESS);

}

  void broadcast(pthread_cond_t *cond)
{
    while(1) {
      int i = 0;
      sem_getvalue(&THERE, &i);
      if (0 == i)
        pthread_cond_broadcast(cond);
      sleep(1);
    }
  }
```

Listing I.3: table.c⟶create_sticks

```
void create_sticks(size_t n)
{
  /* initialize memory for csticks */
  if (NULL == (csticks_l = malloc(n * sizeof(sem_t))))
  {
    perror("could not initialize csticks");
    exit(EXIT_FAILURE);
  }

  for (size_t i = 0; i < n; i++) {
    sem_init(&csticks_l[i], 0, 1);
  }
}
```

Listing I.4: table.c⟶create_philosophs

```
void create_philosoph(size_t number_of_philosophs, size_t age)
{
  /* get the memory we need */
  if (NULL == (philosoph_l = malloc(number_of_philosophs * sizeof(Philosoph_t))))
  {
    perror("could not initialize philosoph");
    exit(EXIT_FAILURE);
  }

  /* initialize the struct with static data */
  for (size_t i = 0; i < number_of_philosophs; i++) {
    philosoph_l[i].id = i;
    philosoph_l[i].age = age;
    philosoph_l[i].times_eaten = 0;
    size_t t = (i -1 + number_of_philosophs ) % number_of_philosophs;
    philosoph_l[i].cstick_l = &csticks_l[t];
    t = (i +1 + number_of_philosophs ) % number_of_philosophs;
```

```
    philosoph_l[i].cstick_r = &csticks_l[t];
    pthread_rwlock_init(&philosoph_l[i].statuslock, NULL);
    philosoph_l[i].status = thinking;
    philosoph_l[i].ISHALLLIVE = 1;

  }

  /* initialize the dynamic data */
  for (size_t i = 0; i < number_of_philosophs; i++) {
    size_t t = (i -1 + number_of_philosophs) % number_of_philosophs;
    philosoph_l[i].left = &philosoph_l[t];
    t = (i +1 + number_of_philosophs) % number_of_philosophs;
    philosoph_l[i].right = &philosoph_l[t];
  }
}
```

Listing I.5: philosoph.c⟶live

```
void live(pPhilosoph_t p)
{
  /* wait of the wakeup call */
  while (-1 == sem_trywait(&THERE) )
  {
    //perror("trylock"); // DEBUG
  }

  printf("Philosoph␣%"PRIu64"␣waiting␣to␣live...\n␣its␣neigbours␣are␣%"PRIu64"␣(l)␣an
  pthread_mutex_lock(&wakeup);
  pthread_cond_wait(&cond, &wakeup);
  pthread_mutex_unlock(&wakeup);

  /* setup the live of Philosoph p */
  srand(time(NULL));

  printf("Philosoph␣%"PRIu64"␣is␣born\n", p->id);
  /* the live of an Philosoph is only thinking, eating and sleeping hungry */
  while (p->ISHALLLIVE)
  {
      think(p);
      eat(p);

     /*if we are death in the next iteration */
    if (0 == --p->age)
    {
      /* put down the sticks */
      /* and change status to deceased */
      pthread_rwlock_wrlock(&p->statuslock);
      p->status = deceased;
      sem_post(p->cstick_l);
      sem_post(p->cstick_r);
```

```
      pthread_rwlock_unlock(&p->statuslock);
      p->ISHALLLIVE = 0;
      printf("Philosoph %"PRIu64" died\n", p->id);
    }
  }
}
```

Listing I.6: philosoph.c⟶eat

```
void eat(pPhilosoph_t p)
{
  /* blocking :: change status to hungry */
  pthread_rwlock_wrlock(&p->statuslock);
  p->status = hungry;
  /* print the status */
  print_status(&p->status, p->id);
  pthread_rwlock_unlock(&p->statuslock);


  /* try to acquire both chopsticks */
  do {

    /* flag if we got the left stick */
    /* if we acquired the left stick while neigbour was 'thinking'
     * and has a lower priority if 'hungry' we need to release the
     * left stick if we cannot acquire the right stick
     */
    uint64_t lstick = 0;

    /* ask the LEFT neigbour about its status */
    /* and try to take the left chopstick     */

    /* blocking :: don't let LEFT change its status */
    pthread_rwlock_rdlock(&p->left->statuslock);
    printf("Philosoph %"PRIu64" checks left neigbour\n",p->id);
    /* check the status */
    switch (p->left->status) {
      case thinking:
      case deceased:
        printf("... left neigbour of Philosoph %"PRIu64" is thinking\n", p->id);
        /* no desire to eat we can get its chopstick */
        sem_wait(p->cstick_l);
        lstick = 1;
        break;
      case hungry:
        printf("... left neigbour of Philosoph %"PRIu64" is hungry\n", p->id);
        if (1 == is_prio_lower(p,p->left)) /* our prio is lower we can eat first */
        {
          printf("... but %"PRIu64" can eat first\n", p->id);
          sem_wait(p->cstick_l);
```

```
        break;
        /* we do not need to set 'lstick' because we would be first */
      }
      /* if we cannot eat first we need to try again */
      pthread_rwlock_unlock(&p->left->statuslock);
      continue;
    case eating:
      printf("... left neigbour of Philosoph %"PRIu64" is eating\n", p->id);
      /* no chance we get that chopstick now, try again */
      pthread_rwlock_unlock(&p->left->statuslock);
      continue; /* repeat the do-while loop from here */
      break;    /* ************ */
    default:    /* never reached */
      break;    /* ************ */
  } /* end of switch */
  pthread_rwlock_unlock(&p->left->statuslock);

  /*****************************************************************************/

  /* ask the RIGHT neigbour about its status */
  /* and try to take the right chopstick     */

  /* blocking :: don't let RIGHT change its status */
  pthread_rwlock_rdlock(&p->right->statuslock);
  printf("Philosoph %"PRIu64" checks right neigbour\n",p->id);
  /* check the status */
  switch (p->right->status) {
    case thinking:
    case deceased:
      printf("... right neigbour of Philosoph %"PRIu64" is thinking\n", p->id);
      /* no desire to eat we can get its chopstick */
      sem_wait(p->cstick_r);
      /* if we are here we already have the left stick */
      /* we can eat now */
      /* lock our status and change it to 'eating' */
      /* this will break the do-while loop         */
      pthread_rwlock_wrlock(&p->statuslock);
      p->status = eating;
      pthread_rwlock_unlock(&p->statuslock);
      break;
    case hungry:
      printf("... right neigbour of Philosoph %"PRIu64" is hungry\n", p->id);
      if (1 == is_prio_lower(p,p->right)) /* our prio is lower we can eat first */
      {
        sem_wait(p->cstick_r);
        /* we already have the left stick and can eat now */
        pthread_rwlock_wrlock(&p->statuslock);
        p->status = eating;
        pthread_rwlock_unlock(&p->statuslock);
```

```
          break; /* leave the do-while loop here */
        }
        /* we did not get right stick
         * so we have to put the left one down
         * in case LEFT shall eat before we can do */
        if (1 == lstick)
        {
          sem_post(p->cstick_l);
        }
        break;
      case eating:
        printf("... right neigbour of Philosoph %"PRIu64" is eating\n", p->id);
        /* no chance we get the right chopstick now,
         * lay down the left one and try again */
        if (1 == lstick)
        {
          sem_post(p->cstick_l);
        }
        pthread_rwlock_unlock(&p->right->statuslock);
        continue; /* repeat the do-while loop from here */
        break;    /* ************ */
      default:    /* never reached */
        break;    /* ************ */
    } /* end of switch */
    pthread_rwlock_unlock(&p->right->statuslock);

  } while (eating != p->status);

  /* we are eating  :: status = eating */
  /* so let's increment 'times_eaten' */
  /* to do it here vs. think()        */
  /* does not complicate things at startup */
  pthread_rwlock_wrlock(&p->priolock);
  p->times_eaten++;
  pthread_rwlock_unlock(&p->priolock);

  /* print the status */
  print_status(&p->status, p->id);

} /* end of eat() */
```

Listing I.7: philosoph.c⟶think

```
void think(pPhilosoph_t p)
{
   /* first change our status to thinking */
  pthread_rwlock_wrlock(&p->statuslock);
  p->status = thinking;
  /* and while no one can read our status
   * lay down chopsticks */
```

```
  sem_post(p->cstick_l);
  sem_post(p->cstick_r);
  pthread_rwlock_unlock(&p->statuslock);

  /* print the status */
  print_status(&p->status, p->id);

} /* end of think() */

/* HELPER FUNCTIONS */
uint64_t is_prio_lower(pPhilosoph_t me, pPhilosoph_t other)
{
  if (me->times_eaten == other->times_eaten)
  {
    return me->id < other->id;
  }
  return me->times_eaten < other->times_eaten;
}
```

Listing I.8: output1

```
Philosoph 0 waiting to live...
 its neigbours are 4 (l) and 1 (r)
Number of Philosophs: 5
Philosoph 4 waiting to live...
 its neigbours are 3 (l) and 0 (r)
Philosoph 3 waiting to live...
 its neigbours are 2 (l) and 4 (r)
Philosoph 1 waiting to live...
 its neigbours are 0 (l) and 2 (r)
Philosoph 2 waiting to live...
 its neigbours are 1 (l) and 3 (r)
Philosoph 0 is born
Philosoph 3 is born
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is thinking
```

```
Philosoph Nr. 3 is hungry
Philosoph Nr. 0 is thinking
Philosoph 3 checks left neigbour
Philosoph Nr. 0 is hungry
... left neigbour of Philosoph 3 is thinking
Philosoph 0 checks left neigbour
... left neigbour of Philosoph 0 is thinking
Philosoph 0 checks right neigbour
Philosoph 3 checks right neigbour
... right neigbour of Philosoph 0 is thinking
... right neigbour of Philosoph 3 is thinking
Philosoph Nr. 0 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is hungry
Philosoph Nr. 0 is eating
Philosoph 3 checks left neigbour
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
... left neigbour of Philosoph 3 is thinking
Philosoph Nr. 0 is thinking
Philosoph 3 checks right neigbour
Philosoph Nr. 0 is hungry
... right neigbour of Philosoph 3 is thinking
Philosoph 0 checks left neigbour
... left neigbour of Philosoph 0 is thinking
Philosoph 0 checks right neigbour
... right neigbour of Philosoph 0 is thinking
Philosoph Nr. 3 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 0 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 3 is thinking
```

```
Philosoph Nr. 0 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is hungry
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is thinking
Philosoph 0 checks left neigbour
Philosoph Nr. 3 is hungry
... left neigbour of Philosoph 0 is thinking
Philosoph 3 checks left neigbour
... left neigbour of Philosoph 3 is thinking
Philosoph 3 checks right neigbour
Philosoph 0 checks right neigbour
... right neigbour of Philosoph 3 is thinking
... right neigbour of Philosoph 0 is thinking
Philosoph Nr. 3 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is hungry
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph 3 checks left neigbour
Philosoph Nr. 0 is thinking
... left neigbour of Philosoph 3 is thinking
Philosoph Nr. 0 is thinking
Philosoph 3 checks right neigbour
Philosoph Nr. 0 is thinking
... right neigbour of Philosoph 3 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 3 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 0 is hungry
Philosoph Nr. 3 is thinking
```

```
Philosoph 0 checks left neigbour
Philosoph Nr. 3 is thinking
... left neigbour of Philosoph 0 is thinking
Philosoph Nr. 3 is thinking
Philosoph 0 checks right neigbour
Philosoph Nr. 3 is thinking
... right neigbour of Philosoph 0 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is hungry
Philosoph 3 checks left neigbour
... left neigbour of Philosoph 3 is thinking
Philosoph 3 checks right neigbour
... right neigbour of Philosoph 3 is thinking
Philosoph Nr. 3 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is hungry
Philosoph Nr. 0 is eating
Philosoph 3 checks left neigbour
Philosoph Nr. 0 is eating
... left neigbour of Philosoph 3 is thinking
Philosoph Nr. 0 is eating
Philosoph 3 checks right neigbour
Philosoph Nr. 0 is eating
... right neigbour of Philosoph 3 is thinking
Philosoph Nr. 0 is eating
```

```
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 0 is thinking
Philosoph Nr. 3 is eating
Philosoph Nr. 0 is thinking
Philosoph Nr. 3 is eating
Philosoph Nr. 0 is thinking
Philosoph Nr. 3 is eating
Philosoph Nr. 0 is hungry
Philosoph Nr. 3 is eating
Philosoph 0 checks left neigbour
... left neigbour of Philosoph 0 is thinking
Philosoph Nr. 3 is eating
Philosoph 0 checks right neigbour
Philosoph Nr. 3 is eating
... right neigbour of Philosoph 0 is thinking
Philosoph Nr. 3 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is hungry
Philosoph Nr. 3 is thinking
Philosoph 0 checks left neigbour
Philosoph Nr. 3 is hungry
... left neigbour of Philosoph 0 is thinking
Philosoph 3 checks left neigbour
... left neigbour of Philosoph 3 is thinking
Philosoph 3 checks right neigbour
... right neigbour of Philosoph 3 is thinking
Philosoph 0 checks right neigbour
Philosoph Nr. 3 is eating
... right neigbour of Philosoph 0 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is thinking
```

```
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is hungry
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph 3 checks left neigbour
Philosoph Nr. 0 is thinking
... left neigbour of Philosoph 3 is thinking
Philosoph Nr. 0 is thinking
Philosoph 3 checks right neigbour
Philosoph Nr. 0 is thinking
... right neigbour of Philosoph 3 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 3 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 0 is thinking
Philosoph Nr. 3 is eating
Philosoph Nr. 0 is hungry
Philosoph Nr. 3 is eating
Philosoph 0 checks left neigbour
... left neigbour of Philosoph 0 is thinking
Philosoph Nr. 3 is eating
Philosoph 0 checks right neigbour
Philosoph Nr. 3 is eating
... right neigbour of Philosoph 0 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is hungry
Philosoph Nr. 0 is eating
Philosoph 3 checks left neigbour
Philosoph Nr. 0 is eating
... left neigbour of Philosoph 3 is thinking
Philosoph Nr. 0 is eating
Philosoph 3 checks right neigbour
Philosoph Nr. 0 is eating
... right neigbour of Philosoph 3 is thinking
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 3 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is thinking
```

```
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is thinking
Philosoph Nr. 3 is hungry
Philosoph 3 checks left neigbour
... left neigbour of Philosoph 3 is thinking
Philosoph 3 checks right neigbour
... right neigbour of Philosoph 3 is thinking
Philosoph Nr. 3 is eating
Philosoph Nr. 3 is eating
Philosoph 3 died
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is hungry
Philosoph 0 checks left neigbour
... left neigbour of Philosoph 0 is thinking
Philosoph 0 checks right neigbour
... right neigbour of Philosoph 0 is thinking
Philosoph Nr. 0 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is hungry
Philosoph 0 checks left neigbour
... left neigbour of Philosoph 0 is thinking
Philosoph 0 checks right neigbour
... right neigbour of Philosoph 0 is thinking
Philosoph Nr. 0 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 0 is eating
```

```
Philosoph Nr. 0 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is thinking
Philosoph Nr. 0 is hungry
Philosoph 0 checks left neigbour
... left neigbour of Philosoph 0 is thinking
Philosoph 0 checks right neigbour
... right neigbour of Philosoph 0 is thinking
Philosoph Nr. 0 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 0 is eating
Philosoph Nr. 0 is eating
Philosoph 0 died
Philosoph 4 is born
Philosoph Nr. 4 is thinking
Philosoph Nr. 4 is thinking
Philosoph Nr. 4 is thinking
Philosoph Nr. 4 is thinking
Philosoph Nr. 4 is thinking
Philosoph Nr. 4 is thinking
Philosoph Nr. 4 is thinking
Philosoph Nr. 4 is hungry
Philosoph 4 checks left neigbour
... left neigbour of Philosoph 4 is thinking
Philosoph 4 checks right neigbour
... right neigbour of Philosoph 4 is thinking
Philosoph Nr. 4 is eating
Philosoph Nr. 4 is eating
Philosoph Nr. 4 is eating
Philosoph Nr. 4 is eating
Philosoph Nr. 4 is eating
Philosoph Nr. 4 is eating
Philosoph Nr. 4 is eating
Philosoph Nr. 4 is eating
Philosoph Nr. 4 is eating
Philosoph Nr. 4 is eating
```

```
Philosoph Nr. 4 is thinking
Philosoph Nr. 4 is thinking
Philosoph Nr. 4 is thinking
Philosoph Nr. 4 is thinking
Philosoph Nr. 4 is thinking
Philosoph 1 is born
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is hungry
Philosoph 1 checks left neigbour
... left neigbour of Philosoph 1 is thinking
Philosoph 1 checks right neigbour
... right neigbour of Philosoph 1 is thinking
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is hungry
Philosoph 1 checks left neigbour
... left neigbour of Philosoph 1 is thinking
Philosoph 1 checks right neigbour
... right neigbour of Philosoph 1 is thinking
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is hungry
Philosoph 1 checks left neigbour
... left neigbour of Philosoph 1 is thinking
Philosoph 1 checks right neigbour
... right neigbour of Philosoph 1 is thinking
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is hungry
```

```
Philosoph 1 checks left neigbour
... left neigbour of Philosoph 1 is thinking
Philosoph 1 checks right neigbour
... right neigbour of Philosoph 1 is thinking
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is hungry
Philosoph 1 checks left neigbour
... left neigbour of Philosoph 1 is thinking
Philosoph 1 checks right neigbour
... right neigbour of Philosoph 1 is thinking
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is hungry
Philosoph 1 checks left neigbour
... left neigbour of Philosoph 1 is thinking
Philosoph 1 checks right neigbour
... right neigbour of Philosoph 1 is thinking
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is hungry
```

```
Philosoph 1 checks left neigbour
... left neigbour of Philosoph 1 is thinking
Philosoph 1 checks right neigbour
... right neigbour of Philosoph 1 is thinking
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is hungry
Philosoph 1 checks left neigbour
... left neigbour of Philosoph 1 is thinking
Philosoph 1 checks right neigbour
... right neigbour of Philosoph 1 is thinking
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is hungry
Philosoph 1 checks left neigbour
... left neigbour of Philosoph 1 is thinking
Philosoph 1 checks right neigbour
... right neigbour of Philosoph 1 is thinking
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is thinking
```

```
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is thinking
Philosoph Nr. 1 is hungry
Philosoph 1 checks left neigbour
... left neigbour of Philosoph 1 is thinking
Philosoph 1 checks right neigbour
... right neigbour of Philosoph 1 is thinking
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph Nr. 1 is eating
Philosoph 1 died
Philosoph 2 is born
Philosoph Nr. 2 is thinking
Philosoph Nr. 2 is thinking
Philosoph Nr. 2 is thinking
Philosoph Nr. 2 is thinking
Philosoph Nr. 2 is thinking
Philosoph Nr. 2 is thinking
Philosoph Nr. 2 is hungry
Philosoph 2 checks left neigbour
... left neigbour of Philosoph 2 is thinking
Philosoph 2 checks right neigbour
... right neigbour of Philosoph 2 is thinking
Philosoph Nr. 2 is eating
Philosoph Nr. 2 is eating
Philosoph Nr. 2 is thinking
Philosoph Nr. 2 is thinking
Philosoph Nr. 2 is thinking
Philosoph Nr. 2 is thinking
Philosoph Nr. 2 is thinking
Philosoph Nr. 2 is hungry
Philosoph 2 checks left neigbour
... left neigbour of Philosoph 2 is thinking
Philosoph 2 checks right neigbour
... right neigbour of Philosoph 2 is thinking
Philosoph Nr. 2 is eating
Philosoph Nr. 2 is eating
Philosoph Nr. 2 is eating
Philosoph Nr. 4 is thinking
```

```
Philosoph Nr. 4 is thinking
Philosoph Nr. 4 is thinking
Philosoph Nr. 4 is thinking
Philosoph Nr. 4 is hungry
Philosoph 4 checks left neigbour
... left neigbour of Philosoph 4 is thinking
Philosoph 4 checks right neigbour
... right neigbour of Philosoph 4 is thinking
Philosoph Nr. 4 is eating
Philosoph Nr. 4 is thinking
Philosoph Nr. 2 is eating
Philosoph Nr. 4 is thinking
Philosoph Nr. 2 is eating
Philosoph Nr. 2 is eating
Philosoph Nr. 2 is eating
Philosoph Nr. 4 is thinking
Philosoph Nr. 2 is eating
Philosoph Nr. 4 is thinking
Philosoph Nr. 2 is eating
Philosoph Nr. 4 is thinking
Philosoph Nr. 2 is thinking
Philosoph Nr. 2 is thinking
Philosoph Nr. 4 is thinking
Philosoph Nr. 2 is thinking
Philosoph Nr. 4 is thinking
Philosoph Nr. 2 is thinking
Philosoph Nr. 4 is thinking
Philosoph Nr. 2 is thinking
Philosoph Nr. 4 is thinking
Philosoph Nr. 2 is hungry
Philosoph Nr. 4 is hungry
Philosoph 2 checks left neigbour
Philosoph 4 checks left neigbour
... left neigbour of Philosoph 2 is thinking
... left neigbour of Philosoph 4 is thinking
Philosoph 2 checks right neigbour
Philosoph 4 checks right neigbour
... right neigbour of Philosoph 4 is thinking
... right neigbour of Philosoph 2 is thinking
Philosoph Nr. 4 is eating
Philosoph Nr. 2 is eating
Philosoph Nr. 2 is eating
Philosoph Nr. 4 is eating
Philosoph Nr. 2 is eating
Philosoph Nr. 4 is thinking
Philosoph Nr. 2 is eating
Philosoph Nr. 4 is thinking
Philosoph Nr. 2 is eating
Philosoph Nr. 4 is thinking
```

```
Philosoph Nr. 2 is eating
Philosoph Nr. 4 is thinking
Philosoph Nr. 2 is eating
Philosoph Nr. 4 is thinking
Philosoph Nr. 2 is eating
Philosoph Nr. 4 is thinking
Philosoph Nr. 2 is thinking
Philosoph Nr. 2 is thinking
Philosoph Nr. 4 is hungry
Philosoph Nr. 2 is hungry
Philosoph 4 checks left neigbour
... left neigbour of Philosoph 4 is thinking
Philosoph 4 checks right neigbour
Philosoph 2 checks left neigbour
... right neigbour of Philosoph 4 is thinking
... left neigbour of Philosoph 2 is thinking
Philosoph Nr. 4 is eating
Philosoph 2 checks right neigbour
Philosoph Nr. 4 is eating
... right neigbour of Philosoph 2 is thinking
Philosoph Nr. 4 is thinking
Philosoph Nr. 2 is eating
Philosoph Nr. 4 is thinking
Philosoph Nr. 2 is eating
Philosoph Nr. 4 is thinking
Philosoph Nr. 2 is eating
Philosoph Nr. 4 is thinking
Philosoph Nr. 2 is eating
Philosoph Nr. 4 is thinking
Philosoph Nr. 2 is eating
Philosoph Nr. 4 is thinking
Philosoph Nr. 2 is thinking
Philosoph Nr. 2 is thinking
Philosoph Nr. 4 is thinking
```

II Verklemmung (8 Punkte)

Bei unserer Implementierung kann es nicht zu Verklemmungen kommen, zusätzlich ist unsere Implementierung fair.
Der Scheduler des Betriebssystems kann jedoch zu anderen Ergebnissen führen. U.a. eine komplett sequenzelle Abarbeitung oder Live bzw. Deadlock.
Wir simulieren die Stäbchen jeweils durch binäre Semaphoren, womit wir eine unteibare Ressource simulieren können. Um eine Endlosschleife zu vermeiden, haben die Philosophen eine maximale Lebenserwartung und altern, was unweigerlich zu ihrem Tode führt und das Programm terminiert. Weiterhin dürfen Sitznachbarn wärend der hier beschriebenen Überprüfung ihren Zustand nicht ändern, um negative Effekte der Nebenläufigkeit auszuschließen (hierzu nutzen wir die pthread_rwlock Funktionen, welche auch die Nutzung der gcc flag $-std = gnu99$ erfordern).

(1) Verklemmungsfreiheit

Im folgenden werden die Begriffe Deadlock und Livelock verwendet, um zwei Arten der Verklemmung unterscheiden zu können.

(1.1) Freedom of Deadlocks

Ein Deadlock liegt vor, wenn wenn alle aktiven Ausführungsfäden auf eine Ressource warten, diese aber nicht freigegeben werden kann (Z.B. weil ein wartender Prozess sie hält. Bei den Philosophen wäre ein solcher Zustand, wenn alle Philosophen das linke Stäbchen hielten.).

Um solche Zustände zu vermeiden, verkünden die Philosophen bevor sie mit dem Aufheben der Stäbchen beginnen, dass sie hungrig sind. Möchte ein hunriger Philosoph essen, überprüft er zunächst, ob seine beiden Nachbarn nicht hungrig sind und nicht essen. Ist dies der Fall, kann er ohne Probleme mit dem Essen beginnen (die Stäbchen aufnehmen). Ist mindestens einer seiner Nachbarn am Essen, wartet der Philosoph bis dies nicht mehr der Fall ist und beginnt erst dann zu essen.

Somit werden nie einzelne Stäbchen aufgenommen und es kann kein Deadlock entstehen.

(1.2) Freedom of Livelock

Ein Livelock liegt vor, wenn nicht alle Ausführungsfäden auf eine Ressource warten, aber dennoch kein Vortschritt statt findet. So könnten in unserem Beispiel Philosophen immer wieder zyklisch Stäbchen aufheben und ablegen ohne jemals zum essen zu kommen. Auf Grund der unter 'Freedom of Deadlocks' beschriebenen Architektur, ist dies jedoch nicht möglich.

(2) Gerechtigkeit

In Bezug auf Gerechtigkeit haben wir darauf geachtet, dass kein Philosoph, der weniger oft gegessen hat als mindestens einer seiner Nachbarn, auf diesen warten muss. Haben mehrere Nachbarn gleich oft gegessen, isst immer der älteste Philosoph zuerst (dies ist nötig, damit die Implementierung der Fairness auch verklemmungsfrei ist).

III Verklemmungsvermeidung (10 Punkte)

Siehe Aufgabe 2.