Musterlösungen zu Übung 9 06.07.2011, Hans-Georg Eßer

Übung 9

Die Lösungsdateien liegen in `producer-consumer-loesungen.zip`.

20 a)
Die wesentliche Änderung im Code ist, dass über zwei Makros `NO_READERS` und `NO_WRITERS` (number of readers/writers) festgelegt wird, wie viele Prozesse der jeweiligen Art es gibt; bei der Initialisierung werden dann entsprechend viele Prozesse erzeugt und später auch wieder eingesammelt. Für die Verwaltung nehmen wir ein Array von Thread-IDs.

Im Detail:
Änderung am Anfang des Programms:

```
#define NO_READERS 3
#define NO_WRITERS 3
```

Änderung in `main()`:

```
main() {
  pthread_t consumer_thread[NO_READERS];  // Thread-Variablen fuer Consumer ...
  pthread_t producer_thread[NO_WRITERS];  // ... und Producer

...

  // Threads erzeugen
  int i;
  for ( i=0; i < NO_WRITERS; i++ ) {
    pthread_create( &producer_thread[i], NULL, (void*)&producer, NULL);
  }
  usleep(100);
  for ( i=0; i < NO_READERS; i++ ) {
    pthread_create( &consumer_thread[i], NULL, (void*)&consumer, NULL);
  }
  sleep(2); printf("\n\nEndwerte:\n");
  list();

  // Aufraeumen:
  for ( i=0; i < NO_WRITERS; i++ ) {
    pthread_join( producer_thread[i], NULL );
  }
  for ( i=0; i < NO_READERS; i++ ) {
    pthread_join( consumer_thread[i], NULL );
  }
```

20 b)
Die Funktion `list()` soll in eine Protokolldatei schreiben, deren Namen legen wir am Anfang des Programms über ein Makro fest (`#define LOGFILE "output.dat"`). Die Ausgabe habe ich in der Musterlösung (`reader-writer-log.c`) etwas übersichtlicher formatiert; in einem Beispiellauf mit je drei Consumern und Producern sowie einem Puffer der Länge 20 erscheint dann folgende Ausgabe:

```
Log-Eintraege:
#1 = mutex
#2 = empty
```

```
#3 = full
#4 = next_read
#5 = next_write
ID | #1 | #2 | #3 | #4 | #5 | Puffer
---+----+----+----+----+----+--------------
 3 |  1 | 12 |  8 |  1 |  9 | [Das ist d__________]
 1 |  1 | 13 |  7 |  2 |  9 | [Das ist d__________]
 2 |  1 | 14 |  6 |  3 |  9 | [Das ist d__________]
 2 |  1 | 12 |  8 |  4 | 12 | [Das ist die _______]
 1 |  1 | 13 |  7 |  5 | 12 | [Das ist die _______]
 3 |  1 | 14 |  6 |  6 | 12 | [Das ist die _______]
 1 |  1 | 12 |  8 |  7 | 15 | [Das ist die Ein____]
 2 |  1 | 13 |  7 |  8 | 15 | [Das ist die Ein____]
 3 |  1 | 14 |  6 |  9 | 15 | [Das ist die Ein____]
 2 |  1 | 12 |  8 | 10 | 18 | [Das ist die Eingan__]
 1 |  1 | 13 |  7 | 11 | 18 | [Das ist die Eingan__]
 3 |  1 | 14 |  6 | 12 | 18 | [Das ist die Eingan__]
 1 |  1 | 12 |  8 | 13 |  1 | [nas ist die Eingangs]
 2 |  1 | 13 |  7 | 14 |  1 | [nas ist die Eingangs]
 3 |  1 | 14 |  6 | 15 |  1 | [nas ist die Eingangs]
 1 |  1 | 12 |  8 | 16 |  4 | [nachist die Eingangs]
 2 |  1 | 13 |  7 | 17 |  4 | [nachist die Eingangs]
 3 |  1 | 14 |  6 | 18 |  4 | [nachist die Eingangs]
 1 |  1 | 12 |  8 | 19 |  7 | [nachric die Eingangs]
 2 |  1 | 13 |  7 |  0 |  7 | [nachric die Eingangs]
 3 |  1 | 14 |  6 |  1 |  7 | [nachric die Eingangs]
 2 |  1 | 12 |  8 |  2 | 10 | [nachricht,e Eingangs]
```

Ich rufe dabei die `list()`-Funktion in jedem Consumer-Thread nach dem Konsumieren auf und gebe zur besseren Übersicht auch die Thread-ID (1 bis `NO_READERS`) aus. So kann man gut sehen, wie die Lese-Threads aus dem Puffer lesen.

Eigentlich wollte ich `list()` aus dem Hauptprogramm heraus aufrufen, während die anderen Threads arbeiten, das hat aber vom Timing her nicht gut funktioniert.