Bibliography

Question

How do you cite a journal article?

Bibliography

Question

How do you cite a journal article?

- The names of the authors (for example, all separated by commas apart from the last two which are separated by “and”).
- The title of the article.
- *The title of the journal*, volume(number): pages, month, year.

Bibliography

Question

How do you cite a journal article?

- The names of the authors (for example, all separated by commas apart from the last two which are separated by “and”).
- The title of the article.
- *The title of the journal*, volume(number): pages, month, year.

Example

Carla Schlatter Ellis. Concurrent Search and Insertion in AVL Trees. *IEEE Transactions on Computers*, 29(9):811–817, September 1980.

Bibliography

Question

How do you record a journal article in BiBTeX?

```
@article{Ellis80,
  author  = "Carla Schlatter Ellis",
  title   = "Concurrent Search and Insertion in
             {AVL} Trees",
  journal = "IEEE Transactions on Computers",
  volume  = "29",
  number  = "9",
  pages   = "811--817",
  month   = sep,
  year    = "1980"}
```

Bibliography

Question

How do you cite a paper in a conference proceedings?

Bibliography

Question

How do you cite a paper in a conference proceedings?

- The names of the authors (for example, all separated by commas apart from the last two which are separated by “and”).
- The title of the paper.
- In, the names of the editors of the proceedings (for example, all separated by commas apart from the last two which are separated by “and”), *the title of the proceedings*, volume of *title of the series*, pages, location where the conference was held, month, year.
- Publisher.

Bibliography

Example

Gordon Plotkin and John Power. Notions of computation determine monads. In, Mogens Nielsen and Uwe Engberg, editors, *Proceedings of the 5th International Conference on Foundations of Software Science and Computation Structures*, volume 2303 of *Lecture Notes in Computer Science*, pages 342–356, Grenoble, France, April 2002. Springer-Verlag.

Bibliography

Question

How do you record a paper in a conference proceedings in BiBTeX?

```
@inproceedings{PlotkinPower02,
  author       = "Gordon Plotkin and John Power",
  title        = "Notions of computation determine
  editor       = "Mogens Nielsen and Uwe Engberg",
  booktitle    = "Proceedings of the 5th Internatio
  volume       = "2303",
  series       = "Lecture Notes in Computer Science
  pages        = "342--356",
  address      = "Grenoble, France",
  month        = apr,
  year         = "2002",
  publisher    = "Springer-Verlag"}
```

The Producer-Consumer Problem

The producer-consumer problem (also known as the bounded-buffer problem) is a classical concurrency problem.

The problem is to synchronize two threads, the producer and the consumer, who share a common, fixed-size buffer. The producer repeatedly generates a data item and puts it into the buffer. At the same time, the consumer removes data items from the buffer, one item at a time.

The problem is to make sure that the producer will not try to add data items to a full buffer and that the consumer will not try to remove data items from an empty buffer.

The Producer-Consumer Problem

We assume that the items are integers. We represent the buffer by means of an array of integers. The array has a fixed size.

```
int N = 10; // capacity of buffer
```

The producer and consumer share the following variables.

```
int[] buffer; // array representing buffer
int next = 0; // index of cell for next item
int size = 0; // number of items stored in buffer
```

The Producer-Consumer Problem

Producer:

```
while (true)
   int value = produce an item;
   buffer[next] = value;
   size++;
   next = (next + 1) mod N;
```

Consumer:

```
while (true)
   int value = buffer[(next - size) mod N];
   size--;
```

The Producer-Consumer Problem

Question

How can we make sure that the producer will not try to add data items to a full buffer?

Question

How can we make sure that the consumer will not try to remove data items from an empty buffer?

The Readers-Writers Problem

The readers and writers problem, due to Courtois, Heymans and Parnas, is another classical concurrency problem. It models access to a database. There are many competing threads wishing to read from and write to the database. It is acceptable to have multiple threads reading at the same time, but if one thread is writing then no other thread may either read or write. The problem is how do you program the reader and writer threads?

The Readers-Writers Problem

The readers and writers share the following variable.

```
semaphore mutex = 1;
```

Reader:

```
P(mutex);
read;
V(mutex);
```

Writer:

```
P(mutex);
write;
V(mutex);
```

The Readers-Writers Problem

Question

Does it solve the readers-writers problem?

The Readers-Writers Problem

Question

Does it solve the readers-writers problem?

Answer

Yes!

The Readers-Writers Problem

Question

Does it solve the readers-writers problem?

Answer

Yes!

Question

Is it a satisfactory solution?

The Readers-Writers Problem

Question

Does it solve the readers-writers problem?

Answer

Yes!

Question

Is it a satisfactory solution?

Answer

No!

The Readers-Writers Problem

Question

Why not?

The Readers-Writers Problem

Question

Why not?

Answer

It does not allow multiple readers to read at the same time.

The Readers-Writers Problem

Options

While a writer is writing, readers and writers arrive. Once the writer is done, can the readers start reading?

Options

While readers are reading, readers and writers arrive. Can the readers start reading?

The Readers-Writers Problem

Options

While a writer is writing, readers and writers arrive. Once the writer is done, can the readers start reading?

Yes

Options

While readers are reading, readers and writers arrive. Can the readers start reading?

Yes

No reader is kept waiting unless a writer is writing.

The Readers-Writers Problem

Options

While a writer is writing, readers and writers arrive. Once the writer is done, can the readers start reading?

Options

While readers are reading, readers and writers arrive. Can the readers start reading?

The Readers-Writers Problem

Options

While a writer is writing, readers and writers arrive. Once the writer is done, can the readers start reading?

No

Options

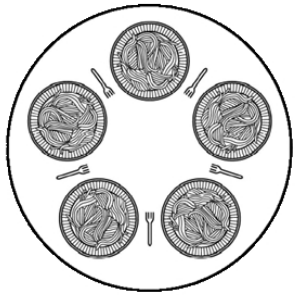
While readers are reading, readers and writers arrive. Can the readers start reading?

No

If a writer wants to write, it writes as soon as possible.

The Dining Philosophers Problem

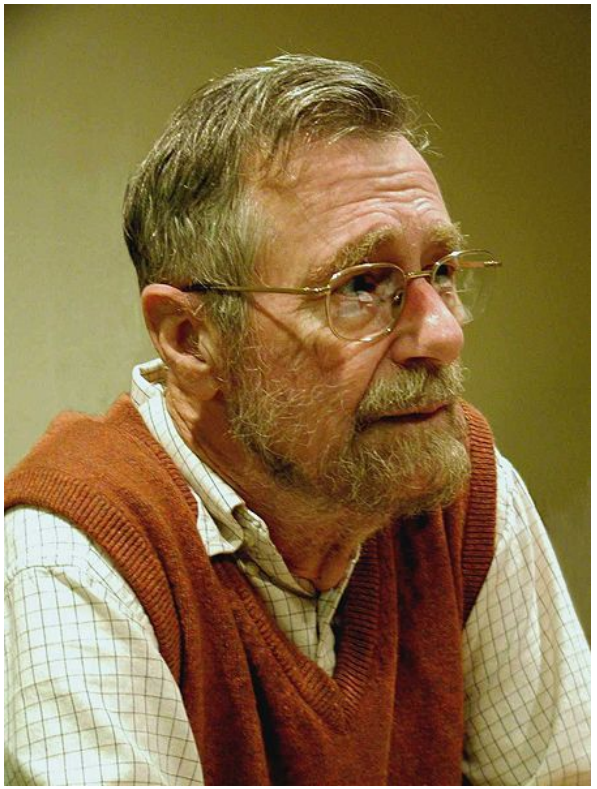
In the dining philosophers problem, due to Dijkstra, five philosophers are seated around a round table. Each philosopher has a plate of spaghetti. The spaghetti is so slippery that a philosopher needs two forks to eat it. The layout of the table is as follows.



The life of a philosopher consists of alternative periods of eating and thinking. When philosophers get hungry, they try to pick up their left and right fork, one at a time, in either order. If successful in picking up both forks, the philosopher eats for a while, then puts down the forks and continues to think.

Edsger Wybe Dijkstra

- Member of the Royal Netherlands Academy of Arts and Sciences (1971)
- Distinguished Fellow of the British Computer Society (1971)
- Recipient of the Turing Award (1972)
- Foreign Honorary Member of the American Academy of Arts and Sciences (1975)



Edsger Wybe Dijkstra

(1930–2002)

The Dining Philosophers Problem

```
int N = 5;

philosopher(i):
while (true)
   think;
   takeForks(i);
   eat;
   putForks(i);

takeForks(i):

putForks(i):
```

The Dining Philosophers Problem

Question

Is this solution correct?

The Dining Philosophers Problem

Question

Is this solution correct?

Answer

No!

The Dining Philosophers Problem

Question

Is this solution correct?

Answer

No!

Question

Why not?

The Dining Philosophers Problem

Question

Is this solution correct?

Answer

No!

Question

Why not?

Answer

Because philosopher(0) and philosopher(1) cat eat at the same time.

The Dining Philosophers Problem

```
int N = 5;
semaphore mutex = 1;

philosopher(i):
while (true)
   think;
   takeForks(i);
   eat;
   putForks(i);

takeForks(i):
P(mutex);

putForks(i):
V(mutex);
```

The Dining Philosophers Problem

Question

Is this solution correct?

The Dining Philosophers Problem

Question

Is this solution correct?

Answer

Yes!

The Dining Philosophers Problem

Question

Is this solution correct?

Answer

Yes!

Question

Does this solution allow two philosophers to eat at the same time?

The Dining Philosophers Problem

Question

Is this solution correct?

Answer

Yes!

Question

Does this solution allow two philosophers to eat at the same time?

Answer

No!

The Dining Philosophers Problem

```
int N = 5;
int state[];
semaphore mutex = 1;

philosopher(i):
while (true)
   think;
   takeForks(i);
   eat;
   putForks(i);
```

The Dining Philosophers Problem

```
takeForks(i):
P(mutex);
if (state[(i − 1) mod N] != EATING
    and state[(i + 1) mod N] != EATING)
  state[i] = EATING;
else
  state[i] = HUNGRY;
V(mutex);

putForks(i):
P(mutex);
state[i] = THINKING;
V(mutex);
```