

School of Electrical Engineering and Computer Science
University of Central Florida
COP5611 - Advanced Operating Systems
Spring 2009 (dcm)

Homework 6. Due Wednesday April 8, 2009.

Consider a circular buffer, in other words $n \geq 1$ buffers: `buf(0)`, `buf(1)`, ... `buf(n-1)` linked in a circular list such that `buf(k)` if preceded by `buf(k-1)` and followed by `buf(k+1)`. A user process \mathcal{U} and a system IO driver \mathcal{S} use two pointers `NextFullBuf` and `NextEmptyBuf` to read the data from the buffer and use it for the computation and to read the data from the disk and fill out a buffer, respectively. The two processes are implemented as coroutines. A coroutine is a program component that generalize subroutines to allow multiple entry points for suspending and resuming execution at certain locations; a subroutine can return only once; in contrast, while a coroutines can return (`yield`) several times. As an example consider two coroutines *put* and *get*:

```
var buf := new queue

coroutine put
  loop
    while buf is not full
      create new items
      add items to buf
    yield to get
  endloop

coroutine get
  loop
    while buf is not empty
      remove items from buf
      use items
    yield to put
  endloop
```

Problem 1 (20 points)

In our example \mathcal{U} yields to \mathcal{S} if the input channel is is not busy and \mathcal{S} yields to \mathcal{U} if the channel is busy. Write a sample code for \mathcal{U} assuming that the program computes without the need to access the data from the file for a while, then uses a procedure `GetBuf(NextFullBuf)` to read an input buffer and use it for computation, then releases the buffer using a procedure `ReleaseBuf` and then the process repeats itself (computes, reads the next buffer, etc.). The code should interspace `yield to S` during its execution. Write also a sample code for \mathcal{S} assuming that initially all buffers are *empty*.

Assume that:

- (i) The initial period in each cycle consists of 4 time units.
- (ii) The computation related to the data in one buffer takes 4 time units.
- (iii) Each procedure call as well as the control statements `repeat`, `until`, `while`, `do`, `yield` each take 1 time unit.
- (iv) The `Read` command issued by \mathcal{S} takes 1 time unit and the channel remains busy for 5 time units.

Using a timing diagram show the cooperation between the two coroutines. Draw separate lines for the main `repeat` loop in \mathcal{U} . Follow the processing of at least three buffers.

Problem 2 (20 points)

Show that \mathcal{U} and \mathcal{S} cooperate correctly:

- (1) The system starts correctly regardless of which coroutine is given initially the control.
- (2) It is impossible for `GetBuf(NextFullBuf)` or \mathcal{S} to loop forever without obtaining a full or empty buffer, respectively.
- (3) The buffers used by `GetBuf(NextFullBuf)` or \mathcal{S} are always the next full buffer and the next empty buffer, respectively.

Problem 3 (20 points)

Consider that each physical record consists of m logical records and the buffers are designated as: `buf(i,j)`, $0 \leq i \leq (n-1)$, $0 \leq j \leq (m-1)$. Assume that operation `Read(ch, Buf(i,*))` will read a physical record into `Buf(i,0)`, `Buf(i,1)`, ..., `Buf(i,m-1)`. Change the code for problem 1 to handle blocked records. Now `GetBuf(NextFullBuffer)` and `ReleaseBuf` are expected to handle logical records.

Problem 4 (20 points)

Assume now that that \mathcal{U} requests m buffers at a time. Rewrite `GetBuf(NextFullBuffer)` and `ReleaseBuf` to accommodate this requirement.

Problem 5 (20 points)

Assume now that a second circular buffer is used for output.

- (1) Write two procedures `GetWriteBuf` and `ReleaseWriteBuf` that request and respectively release write buffers.
- (2) Rewrite \mathcal{S} for two cases: (a) There are two channels, one for input and one for output; (b) the same channel is used for input and output.