COMP 530 Introduction to Operating Systems

Fall 2017  
Kevin Jeffay

Worksheet 11, October 9

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Your Name: |  | You worked with: |  | +1/blank/-1: |  |
|  | Aaron Zhang |  | John Espenhahn |  | +1 |  |
|  |  |  | Brennan Proudfoot |  | +1 |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

1) Show that a general semaphore can be implemented using remote procedure calls (*i.e.*, message passing). Use the blocking (synchronous) primitives defined in the message passing lecture (*i.e.*, select, send, receive, *etc*.).

In more detail, assume shared memory is not supported on some platform and processes synchronize using a distributed semaphore that is implemented by a separate process. User processes that wish to synchronize with each other call functions up and down as before, however these function are now stub functions that send messages to the semaphore process. The semaphore process emulates the behavior of a semaphore in that it blocks the sender of a “down” message if the semaphore’s value is zero, and increments the value of the semaphore on receipt of an “up” message.

Assume the stub functions used by a user process are as shown below. Assume the user processes pass the name of the semaphore (*i.e.*, the name of the semaphore “server” process) to the stub procedure. Lastly, assume that the messages sent and received in the up/down stub functions are not used by the sending or receiving process (*i.e.*, they are effectively null parameters).

|  |  |
| --- | --- |
| procedure up(sem : semaphore)  begin  send(sem, up\_mesg)  receive(sem, results)  end up | procedure down(sem : semaphore)  begin  send(sem, down\_mesg)  receive(sem, results)  end down |

Given these stub functions, give the pseudo-code implementation of the semaphore server process.

Use this space as additional space for answering question 1.

Sender := select()

receive(sender,mesg)

if mesg == up\_mesg:

if !waiting.empty():

send(waiting.pop(),null)

else: sem += 1 send(sender,null)

else if mesg == down\_mesg:

if sem > 0:

sem -= 1

send(sender,null)

else:

waiting.push(sender)