KEY: EXAM ON PROCESSES AND SCHEDULING – SPRING 2005
CSC 262 – OPERATING SYSTEMS
NICHOLAS R. HOWE

1. **Vocabulary** (16 points). Give the technical term that matches the following definitions.

a). This term describes a situation where some set of processes is blocked while waiting for an event that can only be caused by another process in the set.
Deadlock

b). This term refers to a style of interprocess communication where participating processes block until the communication is complete.
Synchronous

c). These terms both refer to a style of process synchronization where processes repeatedly check a variable to see whether they should proceed. (Give two alternate terms.)
Spin-lock, busy waiting

d). This term refers to a scheduling algorithm that must guarantee execution of certain processes by certain pre-specified deadlines.
Real-time or deadline scheduling

e). These terms both refer to an entity stripped to the barest essentials for executing code on a system. (Give two alternate terms.)
Thread, lightweight process

2. **Bounded Buffer** (16 points). This problem has two parts.

a). Consider the following protocol implementing the bounded buffer in a message-passing system, which is missing several key commands. Add the missing commands at appropriate points, choosing from the following options:

```
synch_send(process,message);      // synchronous send
synch_receive(process,message);   // synchronous receive
asynch_send(process,message);     // asynchronous send
asynch_receive(process,message);  // asynchronous receive
```

```
producer:  process
   record rec;
   message m;

   while true do
      produce(rec);
      synch_receive(consumer,m);
      m = build_message(rec);
      asynch_send(consumer,m)
   endwhile;
endprocess;
```

```
consumer:  process
   record rec;
   message m;

   for int i = 1 to N
      asynch_send(producer,m);
   endfor;
   while true do
      synch_receive(producer,m);
      rec = extract_record(m);
      asynch_send(producer,m);
      consume(rec);
   endwhile;
endprocess;
```

b). There is no buffer apparent in the above code, yet it is supposed to be an example of a bounded buffer. Explain where the buffering occurs.

The operating system must implement a buffer to hold the message queue. Messages sent by one process wait in this system buffer until they are received. In some sense, this bounded buffer is simply built on top of another one at a lower level.

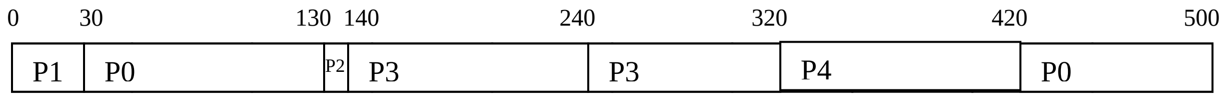
3. **Process Scheduling** (24 points). Consider the process table below, and answer the questions that follow under the following assumptions:

- After it has executed for 30 ms, process P_1 will block.
- After 140 ms (regardless of which process executes), a device interrupt will unblock P_3.
- After it has executed for 180 ms, process P_3 will terminate.
- After 220 ms (regardless of which process executes), a new low-priority process is created with PID 4.
- The "last ran at" field gives the system time in ms at which the process's last execution was interrupted. The current system time is 10000 ms.

PID	0	1	2	3
Priority	Low	High	Low	High
Status	Ready	Ready	Ready	Blocked
Last ran at	9750	9930	9850	10000

a). The kernel uses priority queues for scheduling. If there are multiple processes in a queue, they are scheduled round-robin based upon the last-ran-at entry. The quantum length is 100 ms, and context switching time is negligible (essentially zero). Draw a

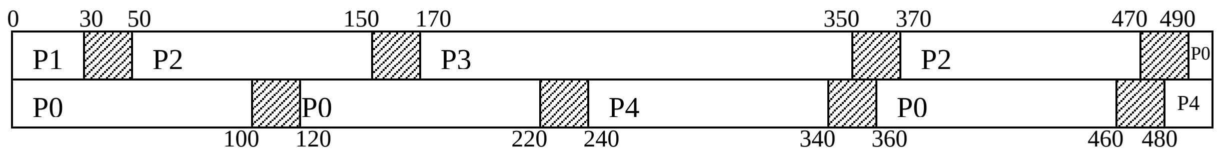
Gantt chart showing the running processes over the next 500 ms. Label the time of each transition.



b). Draw the updated process table as it should appear after 500 ms.

PID	0	1	2	4
Priority	Low	High	Low	Low
Status	Running	Blocked	Ready	Ready
Last ran at	9750	10030	9850	

c). Now assume that there are two CPUs. This time, the kernel uses simple round-robin scheduling, but high-priority processes get an extra-long quantum (200 ms), and context switching takes 20 ms. Draw a new Gantt chart. Label the time of each transition.



4. **Interrupt Sequence** (12 points). This problem has two parts.

a). When an interrupt occurs, how does the kernel determine the cause of the interrupt? Be precise in your answer, referring by name to the pieces of hardware involved.

The piece of hardware causing the interrupt sends a signal to the interrupt controller on a dedicated wire. The interrupt controller then sends a signal to the CPU requesting an interrupt. When the CPU acknowledges the interrupt, the interrupt controller puts a code on the system bus corresponding to the interrupt type. The CPU reads this code off the bus.

b). When a process makes a system call, how does the kernel determine what action is desired by the process?

Before making the system call, the process stores a code describing the desired action in a predetermined register.

5. **Race Conditions and Semaphores** (16 points). For each of the following programs, state whether the program will always terminate, sometimes terminate, or never terminate.

Next, state all the possible values for *x* and *y* if the program terminates. Express your answer as a set of *(x,y)* pairs. For example, the starting configuration should be written as {(3,5)}.

You may assume that the variables are initialized as follows:

x : integer init 2
y : integer init 3
s : general semaphore init 1
t : general semaphore init 1

a). `cobegin x := y; // y := x; coend`
Always terminates; possible values {(3,2),(3,3),(2,2)}

b).
```
cobegin
    DOWN(s); x := y; UP(s);
//
    DOWN(s); y := x; UP(s);
coend
```
Always terminates; possible values {(3,3),(2,2)}

c).
```
cobegin
    DOWN(s); x := y; UP(s);
//
    DOWN(t); y := x; UP(t);
coend
```
Always terminates; possible values {(3,2),(3,3),(2,2)}

d).
```
cobegin
    DOWN(s); x := y; UP(s); UP(s);
//
    DOWN(s); DOWN(s); y := x; UP(s);
coend
```
Sometimes terminates; possible values { (3,3)}

6. **Deadlock** (16 points). The D. Edwin Locke Memorial Library has unusal lending policies. Patrons may take out books on indefinite loan, and the library has no recall mechanism in place. Fortunately, the patrons are very conscientious and always return books they no longer need. Nevertheless, there have been complaints recently of book hoarding on the part of some patrons. When the library investigates, the patrons accused of hoarding claim that they are working on a project and need one more book before they can finish it and return everything. As soon as someone else returns the books they need, they will finish their project and return all the books they have out.

a). Is it possible to reach a state of deadlock with these policies? State the conditions necessary for deadlock and whether or not each one applies in this case.

It is possible to reach deadlock. Conditions:

1. Mutually exclusive access to resources: applies
2. Partial allocation possible: applies
3. Circular waiting possible: applies
4. No preemption: applies

b). A change is proposed to the lending rules. Patrons may still keep books indefinitely, but they are now only allowed to check out one set of books at a time. If they find they need other books, they must first bring back the ones they have out. (Of course, they may renew books at this time if no other patrons are waiting to take them out.) Discuss how this change affects your answer given in part (a) above.

This change voids requirement 2, on partial allocation. Thus deadlock is no longer possible under the proposed policy. (Circular waiting can occur momentarily.)

c). Instead of the above change, a different policy is adopted. All patrons working on a project are required to submit in advance a list of the full set of books they might need to take out. When they need a book, they must request it from a librarian, who may or may not give it to them right away. The library promises, however, that if every patron abides by this new policy, everyone will eventually get the books they have requested. Discuss this policy in relation to your answer to part (a) above. Is it possible that the library can keep its promise?

None of the conditions necessary for deadlock is voided. However, if the librarians adopt a deadlock-avoidance strategy and only make loans that maintain a safe state, then deadlock may be avoided and the library's promise can be kept.