OS test questions and answers

Question 1.

There are two major design types for operating system kernels: monolithic kernels and microkernels. Which of these types better satisfies the following requirements? If both are equally as good write both. Justify your answers.

a) Convenient access to operating system data structures.
Monolithic – access from any part of the kernel to any other part is possible. With a microkernel access is via message passing.

b) Addition of new operating system components.
Microkernel – no (or little) modification is required in the kernel the new service is in user space.

c) Modification of operating system components.
Microkernel – no (or little) modification is required in the kernel. Modifying a user mode service can be done more easily. It should be possible to modify without a need to restart the system.

d) Security and reliability.
Microkernel – services are isolated from each other. If one service crashes it does not directly affect others. Another acceptable answer on security would be – both, if implemented well.

8 marks

Question 2.

In the following Linux program written in C the function `getpid()` returns the process id of the current process.

The line
```
printf("%d. childpid: %d\n", i, childpid);
```
is equivalent to the Java

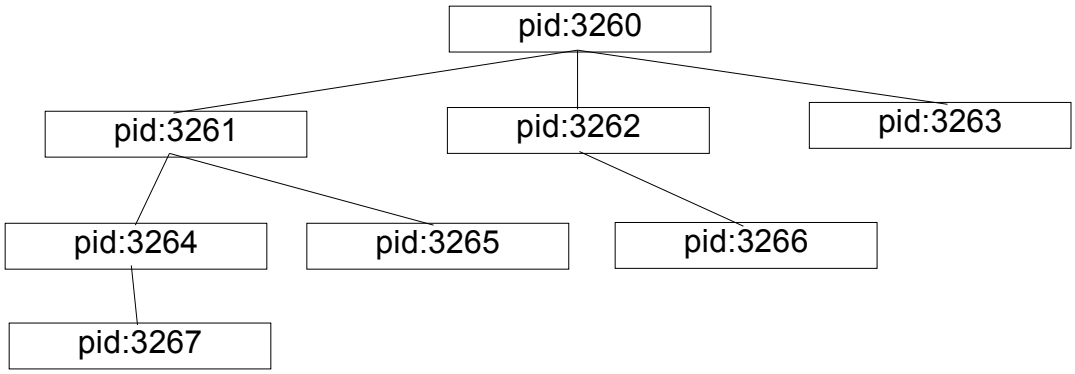
```
System.out.println(i+". childpid: "+childpid);
```

```
/* testForkExample.c */
main() {
        int i = 0;
        printf("main pid: %d\n", getpid());
        while (i < 3) {
                int childpid = fork();
                if (childpid != 0)
                        printf("%d. childpid: %d\n", i, childpid);
                i++;
        }
}
```

The program produced the following output when run from a terminal on my machine:

```
[robert@icicle Documents]$ testForkExample
main pid: 3260
0. childpid: 3261
1. childpid: 3262
2. childpid: 3263
[robert@icicle Documents]$ 1. childpid: 3264
2. childpid: 3265
2. childpid: 3266
2. childpid: 3267
```

a) Draw a process tree showing each of the processes produced with links between parent and child processes. (There is more than one possible correct answer.)



Other acceptable answers have 3265, 3266 and 3267 swapped amongst themselves. The others must be in these places.

6 marks

b) Explain why the last four lines of output are produced after the command line prompt [robert@icicle Documents]$ reappears.

The prompt can reappear as soon as the testForkExample process itself has finished. The shell waits for any process it starts to finish it does not wait for children of that process to finish. All other lines of output come from the children.

2 marks

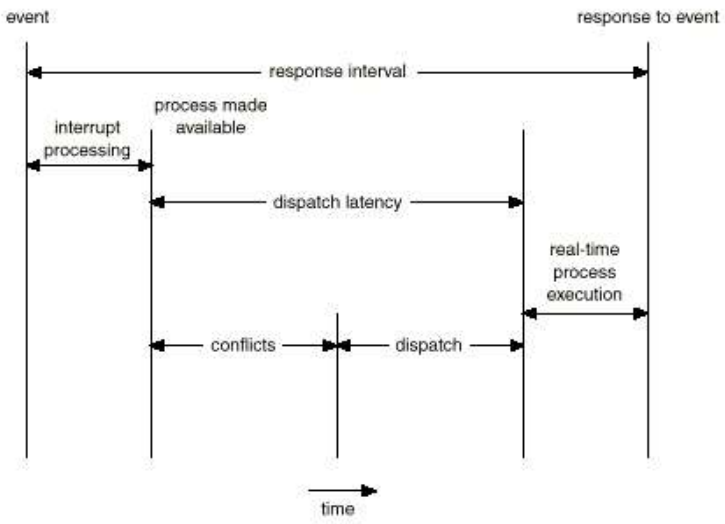
Question 3.

a) In the lectures it was mentioned that priorities make real-time scheduling more difficult. Give a reason why priorities are still used in real-time systems

Priorities can be used to indicate how important tasks are. In this way we can ensure that more important tasks are handled before less important ones.

2 marks

b) Here is a diagram from the textbook:



Briefly describe each of the following terms shown in the diagram:

conflicts	*Conflicts are conditions that cause the required response to be postponed, e.g., some required resource is unavailable and a process needs preempting.*
dispatch	*Making the required response change state to running. At the end of this time the real-time process is running.*
dispatch latency	*The total time from when the correct real-time process is identified by the interrupt handler through to the start of execution of that process.*
response interval	*The total time from when the event occurs until the event has been completely handled.*

8 marks

Question 4.

The Java thread method suspend() is deprecated because it can produce deadlocks. Describe a situation where the use of suspend results in a deadlock.

Any situation where the thread that is going to call resume on the suspended thread itself blocks waiting for a lock currently held by the suspended thread.

5marks

Question 5.

In the early years of digital computers, programmers could work interactively on the machines. For many years following that time programmers had to submit their programs for processing at a later time and could no longer control the machines directly. What changes needed to be made to the hardware and operating systems that allowed programmers safe interactive access to the computers again on time-sharing systems? Also say why the changes were needed.

Memory protection. So that a programmer working on the computer doesn't corrupt memory being used by another user or program.
Protected processor mode. To stop the programmer directly accessing resources that need protection either because they are shared or because they are dangerous if used incorrectly.
Plenty of resources (especially more memory), time-sharing systems cannot maximise resource usage without penalizing usability.
Terminals for all users. Obvious need for interactive computing.
Scheduling needed to change (or be added). Want to provide good service to the user rather than maximising the use of the hardware.
Security. Apart from the hardware changes above, interactive computing allows greater possibilities for security violations, so file system and other security needed adding or improving.
etc.

6 marks

Question 6.

a) A scheduler uses multi-level feedback queues with the following characteristics:

Level	***Time-slice***	***When dispatched***
1	10 msec	
2	40 msec	Only when level 1 is empty
3	Unlimited	Only when levels 1 and 2 are empty

Processes enter the scheduler in the level 1 queue and move down a level if they completely use their time-slices. Processes do not move up levels. Preemption only occurs when the time-slice is completed.

This list of processes shows CPU burst times. The processes start in this order in the level 1 queue.

Process name	***CPU burst time***
A	8
B	300
C	60
D	12
E	5
F	20

Using the information above complete this table showing: the order of process scheduling; the queue the process was selected from; and how long each process runs.

Process	***Queue***	***Run time***
A	1	8
B	*1*	*10*
C	*1*	*10*
D	*1*	*10*
E	*1*	*5*
F	*1*	*10*
B	*2*	*40*
C	*2*	*40*
D	*2*	*2*
F	*2*	*10*
B	*3*	*250*
C	*3*	*10*

5 marks

b) There is something very wrong with this scheduler. What is it and how can it be fixed?

When processes descend to level 3 the system can be locked up by a level 3 process running in an infinite loop.
Fix it by giving level 3 processes a long but fixed time-slice and scheduling them round-robin.

2 marks

Question 7.

The semaphore wait and signal operations are defined as indivisible (or atomic) operations.

a) Why do they have to be indivisble?

If they weren't indivisble it would be possible for multiple waits and signals to occur simultaneously (on a multiprocessor) or at least interleaved with each other. This would corrupt the value of the semaphore and prevent it working properly.

4 marks

b) Because they are indivisible does that mean that all other processes running on a multiprocessor system must stop when a wait or signal operation is executed? Explain why or why not.

No. Strictly only processes accessing the same semaphore at that time need to stop. Any processes not wanting to change or access the semaphore should be allowed to continue.

4 marks

Question 8.

a) Describe a scheme to implement mutual exclusion in a distributed environment. Say how a process requests exclusive access to the resource, how that exclusive access is maintained and how the resource is made available to a new process when the current one has finished with it.

Either a centralized or distributed solution is acceptable. See the notes.

6 marks

b) What are two problems with the scheme you described in a)?

If the site hosting the process with the resource goes down any solution will need to account for that.
In centralized systems the coordinator process going down is a problem.
Network failures will be a problem in any solution.

2 marks