This exam is closed book. You can have one page of notes. UH expels cheaters.

1. Match each of the following features with the single sentence that describes it best: (10×3 points)

   (Hint: Several of the choices offered are plain wrong.)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>lightweight processes</td>
<td>D. Creates a new process.</td>
</tr>
<tr>
<td>timer interrupts</td>
<td>H. Limit the amount of time a process can occupy a core.</td>
</tr>
<tr>
<td>microkernel</td>
<td>E. Delegates most of its duties to user-level servers.</td>
</tr>
<tr>
<td>ready queue</td>
<td>C. Contains all the processes waiting for the CPU.</td>
</tr>
<tr>
<td>privileged mode</td>
<td>A. Allows the CPU to execute input/output instructions.</td>
</tr>
</tbody>
</table>

   A. Allows the CPU to execute input/output instructions.
   B. Contains all processes ready to access a peripheral.
   C. Contains all the processes waiting for the CPU.
   D. Creates a new process.
   E. Delegates most of its duties to user-level servers.
   F. Gives a disk controller direct access to the main memory.
   G. Grants user processes direct access to BIOS routines.
   H. Limit the amount of time a process can occupy a core.
   I. Loads in memory the program to be executed by a given process.
   J. Provides access to distributed memory.
   K. Sends a signal to another process.
   L. Share the address space of their parent.
   M. Specifies a superuser process.
   N. Specifies what a process should do when it receives a signal.
   O. Terminates the process making that system call.
2. *Questions with short answers*: (6×5 points)

A. Why is memory protection always implemented in *hardware*?

   Because it must be done for every memory reference.

B. What is the major disadvantage of the *master-slave organization* in operating systems for multiprocessor architectures?

   Requiring all kernel functions to be performed on a single processor will result in a potential bottleneck.

C. What is the major disadvantage of *single-threaded file servers*?

   They can only process one request at a time.

D. Why should all *multiprocessor operating systems* provide *kernel-supported threads*?

   To allow parallel user programs to run or more than on core at a time.

E. Which processes should we *not suspend* unless we absolutely need to do so?

   The processes in the READY queue, OR high-priority processes OR RT processes.

F. What is the major disadvantage of *user-level threads*?

   The whole set of threads sharing an address space will be blocked each time one of the threads does a blocking system call.
3. Add the two system calls that will ensure that the program will print **Hello World! four times**.
(2*5 points)

```c
#include <stdio.h>
int main(){

    fork();

    fork();
    printf("Hello world!\n");
}
```

4. Why should we **prevent** users of a multi-user system from **rebooting** the OS from a DVD or a flash drive? (10 points)

   It would allow them to load a rogue OS that would bypass all the security features of the system OS.

5. Which transitions can bring a process into the **ready state**? (4*5 points)

   A. From: **the WAITING/BLOCKED state**

      When: **the process completes a system call**.

   B. From: **the RUNNING state**

      When: **the process is preempted OR timer interrupt OR a higher priority process arrives**.

   C. From: **the NEW state**.

      When: **the process arrives**.

   D. From: **the SUSPENDED-READY state**

      When: **the process is reactivated/brought onto main memory**.
This exam is closed book. You can have one page of notes. UH expels cheaters.

1. Match each of the following features with the single sentence that describes it best: (10×3 points)
   (Hint: Several of the choices offered are plain wrong.)

<table>
<thead>
<tr>
<th>Feature</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>signal( )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>O</td>
</tr>
<tr>
<td>microkernel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D</td>
</tr>
<tr>
<td>ready queue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
</tr>
<tr>
<td>privileged mode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>I</td>
</tr>
<tr>
<td>fork( )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C</td>
</tr>
<tr>
<td>execv( )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>K</td>
<td></td>
</tr>
<tr>
<td>kill( )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>timer interrupts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>J</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lightweight processes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>N</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A. Contains all processes ready to access a peripheral.
B. Contains all the processes waiting for the CPU.
C. Creates a new process.
D. Delegates most of its duties to user-level servers.
E. Designates a superuser process.
F. Ends the process making that system call.
G. Gives a disk controller direct access to the main memory.
H. Grants user processes direct access to BIOS routines.
I. Lets the CPU execute input/output instructions.
J. Limit the amount of time a process can occupy a core.
K. Loads in memory the program to be executed by a given process.
L. Provides access to distributed memory.
M. Sends a signal to another process.
N. Share the address space of their parent.
O. Specifies what a process should do when it receives a signal.
2. **Questions with short answers:** (6×5 points)

A. Why should all **multiprocessor operating systems** provide **kernel-supported threads**?

   *To allow parallel user programs to run on more than one core at a time.*

B. What is the major advantage of the **symmetric organization** in operating systems for multiprocessor architectures?

   *Since kernel functions can be performed on any processor, they will not cause bottlenecks.*

C. Why is memory protection always implemented in **hardware**?

   *Because it must be done for every memory reference.*

D. What is the major disadvantage of **single-threaded file servers**?

   *They can only process one request at a time.*

E. What is the major disadvantage of **kernel supported threads**?

   *Switching between two threads of the same task requires two context switches.*

F. What is the main disadvantage of **microkernels**?

   *Each system call that involves a user-level server will require four contexts switches.*
3. Add the two system calls that will ensure that the program will print four lines. (2×5 points)

```c
#include <stdio.h>
int main(){
    fork();
    fork();
    printf("Hello world!\n");
} // main
```

4. Why should we prevent users of a multi-user system from rebooting the OS from a DVD or a flash drive? (10 points)

It would allow them to load a rogue OS that would bypass all the security features of the system OS.

5. Give an example of a real time process with soft deadlines? (5 points)

Paying a video on your PC/ Watching Netflix.

6. Which transitions can take a process from the running state? (3×5 points)

A. To: READY state

When: the process is preempted OR timer interrupt OR a higher priority process arrives.

B. To: WAITING/BLOCKED state

When: the process issues a system call.

C. To: TERMINATED state

When: the process terminates.