**Problem 1 (15pts):**

a) What is the main advantage of the microkernel approach to operating system design?

b) How do user programs and system services interact in a microkernel architecture?

c) What are the disadvantages of using the microkernel?

**Problem 2 (10pts):**

What is the difference between fork() and exec() on UNIX?

**Problem 3 (10pts):**

What resources are used when a thread is created? How do these differ

from those used when a process is created?

**Problem 4 (10pts):**

What are context switches used for and what does a typical context switch involve?

**Problem 5 (18pts):**

As shown below, processes can be in one of three states: *running, ready, blocked*. There are six possible state transitions (labeled 1-6). For each label, indicate whether the transition is valid or not valid. If valid, indicate when the transition is used for a process (i.e. give an example). If the transition is not valid, then indicate why.

1. 1-Blocked to Ready:
2. 2-Ready to Running:
3. 3-Running to Blocked:
4. 4-Ready to Blocked:

1. 5-Running to Ready:
2. 6-Blocked to Running:

**Problem 6 (2+20pts):**

In the code below, assume that:

1. All fork() and execvp() statements execute successfully (no failure).
2. The program arguments of execvp() do not spawn more processes or print out more characters.
3. All pid variables (pid1 … pid6) are initialized to 0.

|  |
| --- |
| void main() {  pid1 = fork();  if (pid1 == 0) {  pid2 = fork();  printf("A");  } else {  execvp(...)  }  printf("B");  pid3 = fork();  if (pid4 != 0) {  printf("C");  execvp(...);  } else {  if (pid1 != 0) {  pid5 = fork();  execvp(...);  printf("D");  }  }  if (pid2 > 0) {  pid6 = fork();  printf("E");  } else {  printf("F");  execvp(...);  }  printf("G");  } |

Now answer the questions on the following page.

1. How many processes will be created by the execution of this code (including the initial process)?
2. In a “process creation diagram” (similar to what we covered in class), show the order in which each process is created , and the values of pid1 to pid6 for each process, as well as the output of each process (characters getting printed).

**Problem 7 (15pts):**

Please mark the correct answer in the following questions.

a) Policy \_\_\_\_.

A) determines how to do something

B) determines what will be done

C) is not likely to change across places

D) is not likely to change over time

b) The major difficulty in designing a layered operating system approach is \_\_\_\_.

A) appropriately defining the various layers

B) making sure that each layer hides certain data structures, hardware, and operations from higher-level layers

C) debugging a particular layer

D) making sure each layer is easily converted to modules

c) \_\_\_\_\_ allow operating system services to be loaded dynamically.

A) Virtual machines

B) Modules

C) File systems

D) Graphical user interfaces

d) A \_\_\_\_\_ uses an existing thread — rather than creating a new one — to complete a task.

A) lightweight process

B) thread pool

C) scheduler activation

D) asynchronous procedure call

e) The \_\_\_\_\_ model maps each user-level thread to one kernel thread.

A) many-to-many

B) two-level

C) one-to-one

D) many-to-one