Sistemas Operativos I - Práctica 1

1. Que es multiprogramación?
2. En las primeras computadoras, la CPU manejaba en forma directa cada byte de datos que se leía y escribía (es decir, no había DMA). Que implicaciones tenía esto para la multiprogramación?
3. Cuales de las siguientes instrucciones deberían ejecutarse en modo protegido (kernel/supervisor)?
   A. Inhabilitar todas las interrupciones
   B. leer el reloj de hora del día
   C. escribir el reloj de hora del día
   D. modificar el mapa de memoria
4. Tenemos una MMU que primero compara una dirección virtual con el registro límite fallando si es superior. En un diseño alternativo primero se suma la dirección virtual al registro base y luego se compara. Son lógicamente equivalentes? y en cuanto a desempeño? Nota: en el primer diseño el registro límite guarda el registro base + límite, en el segundo el límite.
5. Cuando un usuario hace una llamada al sistema para leer o escribir un archivo, da el descriptor del archivo (previamente obtenido con una llamada a open()), el puntero al buffer de datos y la cantidad. Luego se transfiere el control al sistema operativo, que llama al controlador del dispositivo apropiado. Supongamos que el controlador pone en marcha el disco y se suspende hasta que el disco termina y se presenta una interrupción. En el caso de la lectura es obvio que el que invoca tendría que bloquearse (esperando los datos para seguir trabajando!). Que sucedería en el caso de escribir? Es necesario que el invocador se bloquee hasta que termine la transferencia a disco?

   ```
   nread = read(fd, &buffer, nbytes)
   nwritten = write(fd, &buffer, nbytes)
   ```
6. Por qué se necesita la tabla de procesos en un sistema de tiempo compartido?

También se necesita en sistemas de computadoras donde hay un sólo proceso que se adueña de la CPU hasta que termina?

7. De una condición por la cual podría fallar la llamada al sistema fork o exec.
8. En C la llamada de biblioteca se lama read, igual que la llamada de UNIX. Es esto indispensable? Cual es más importante?
9. En la clase vimos una comparación entre llamadas al sistema de UNIX y WIN32 y algunas de UNIX no existían en WIN32. Que implicancias tiene esto para un programador que quiere convertir un programa UNIX para que se ejecute en WIN32?