Quiz laborator 6

1. În Linux, porturile alocate apar în:
   - /proc/ioports
   - /proc/interrupts
   - /proc/iomem
   - /proc/stat
2. Pentru a proteja accesul la date partajate între rutina de tratare a întreruperii şi operaţia `write` a unui device driver se foloseşte:
   - `spin_lock`/`spin_unlock` în rutina de tratare a întreruperii şi `spin_lock`/`spin_unlock` în `write`
   - `spin_lock`/`spin_unlock` în rutina de tratare a întreruperii şi `spin_lock_irqsave`/`spin_unlock_irqrestore` în `write`
   - `spin_lock`/`spin_unlock` în rutina de tratare a întreruperii şi `disable_irq`/`enable_irq` în write
   - `spin_lock`/`spin_unlock` în rutina de tratare a întreruperii şi `local_irq_disable`/`local_irq_enable` în write
3. O întrerupere partajată:
   - se înregistrează prin specificarea flag-ului `IRQF_SHARED` pentru `request_irq`
   - este tratată prin execuţia tuturor rutinelor înregistrate
   - se înregistrează cu `request_irq(MY_IRQ, my_handler, IRQF_SHARED, "dev_name", NULL)`
   - trebuie să verifice registrul de stare în rutina de tratare a întreruperii pentru a determina daca întreruperea a fost generată de dispozitivul pe care îl gestionează
4. Care din următoarele variante sunt corecte?
   - `outb(value, MY_BASEPORT);`
   - `outb(MY_BASEPORT, value);`
   - `WRITE_PORT_UCHAR (value, (PUCHAR) MY_BASEPORT);`
   - `WRITE_PORT_UCHAR ((PUCHAR) MY_BASEPORT, value);`
5. În Linux, putem afla numărul de întreruperi generate din :
   - /proc/ioports
   - /proc/interrupts
   - /proc/iomem
   - /proc/stat
6. Pentru a proteja accesul la date partajate între rutina de a întreruperii şi operaţia `write` a unui device driver se foloseşte:
   - `KeAcquireSpinLock`/`KeReleaseSpinLock` în rutina de tratare a întreruperii şi `KeAcquireSpinLock`/`KeReleaseSpinLock` în `write`
   - `KeAcquireInStackQueuedSpinLock`/`KeReleaseInStackQueuedSpinLock` în rutina de tratare a întreruperii şi `KeAcquireSpinLock`/`KeReleaseSpinLock` în `write`
   - accesul direct la date în rutina de tratare a întreruperii şi `KeSynchronizeExecution` în write
   - `KeSynchronizeExecution` în rutina de tratare a întreruperii şi accesul direct la date în write
7. Pentru activarea întreruperii care apare la primirea de date pe portul serial trebuie setat:
   - bitul 0 (`Enable Received Data Available Interrupt`) în registrul `IER`
   - bitul 0 (`Interrupt Pending`) din registrul `IIR`
   - bitul 0 (`Enable FIFO's`) din registrul `FCR`
   - bitul 3 (`Aux Output 2`) în registrul `MCR`
8. În rutina de tratarea a întreruperii:
   - trebuie dezactivate întreruperile la intrarea în funcţie
   - se verifică registrul de stare pentru a determina dacă întreruperea a fost generată de dispozitivul pe care îl gestionează
   - se întoarce `IRQ_NONE`/`FALSE` dacă întreruperea nu a fost generată de dispozitivul pe care îl gestionează şi `IRQ_HANDLED`/`TRUE` în caz contrar
   - se resetează bitul `interrupt-pending` pe dispozitivul fizic
9. Parametrul `SynchronizeIrql` al funcţiei `IoConnectInterrupt`:
   - specifică valoarea IRQ-ului la care va rula rutina de tratare a înteruperii
   - este folosit pentru sincronizare între rutina de tratare a întreruperii şi contex proces
   - trebuie să fie valoarea maximă dintre toate întreruperile servite de o rutină de tratare a întreruperii
   - trebuie să aibă aceeaşi valoare cu parametrul `Irql` al aceleiaşi funcţii
10. Care din următoarele secvenţe vor eşua la obţinerea unei întreruperi?
    - `request_irq(IRQ, handler1, IRQF_SHARED, "dev1", data1); request_irq(IRQ, handler2, IRQF_SHARED, "dev2", data2);`
    - `request_irq(IRQ, handler1, 0, "dev1", data1); request_irq(IRQ, handler2, IRQF_SHARED, "dev2", data2);`
    - `request_irq(IRQ, handler1, IRQF_SHARED, "dev1", data1); request_irq(IRQ, handler2, 0, "dev2", data2);`
    - `request_irq(IRQ, handler1, 0, "dev1", data1); request_irq(IRQ, handler2, 0, "dev2", data2);`