Identifikation

Wichtig: Bitte tragen Sie hier entweder Ihren Namen ein oder füllen Sie die Felder zur Identifikation (rechte Spalte) aus, damit sich die Fragebögen aus vorherigen Tests diesem Bogen zuordnen lassen. Sie können auch beide Bereiche ausfüllen.

Ihre Testergebnisse verwende ich **nicht** zur Beurteilung, Notenvergabe etc., und die Testergebnisse werden nur in anonymisierter Form öffentlich gemacht.

Name: _____

Vorname: _____

Dritter Buchstabe im Vornamen der Mutter: ___
*z. B.: MA**R**IA MÜLLER → **R***

Letzte Ziffer Ihrer Hausnummer: ___
*z. B.: Hauptstraße 11**9** → **9***

Summe Geb.-Tag + Geb.-Monat: ___
*z. B.: **14.09.**1982 → 14 + 9 = **23***

Anleitung

Auf den folgenden Seiten finden Sie verschiedene Aussagen, die entweder richtig oder falsch sind. Bitte lesen Sie sich jede Aussage durch und setzen Sie ein Kreuz in der Spalte **Ja** oder **Nein**, wenn Sie sich sicher sind – anderenfalls setzen Sie ein Kreuz in der grauen Spalte **Weiß nicht**. Die ersten Fragen sind identisch mit denen von Fragebogen I und II; auf Seite 3 folgen Fragen zur Implementierung der Interrupt-/Fault-/Syscall-Behandlung.

Bei Aussagen zu konkreten Umsetzungen nehmen wir an, dass ein Unix-ähnliches Betriebssystem auf einem klassischen PC (mit Intel-i386-CPU und Intel-8259-Interrupt-Controllern) läuft. Unter „Prozess“ verstehen wir einen normalen Prozess, der im User Mode läuft (keinen „Kernel-Prozess“).

Hatten Sie vor der Vorlesung schon Vorkenntnisse zum Thema Interrupt-Behandlung? ☐ Ja ☐ Nein

Falls ja, dann für welche Hardware-Plattform? _____

Nr.	Ja	Nein	Weiß nicht	Aussage
1	J	N X		Polling ist eine spezielle Form der Interrupt-Behandlung.
2	J X	N		Tritt ein Interrupt auf, während ein Prozess läuft (und Interrupts aktiviert sind), wird dessen Ausführung unterbrochen, um einen Interrupt- Handler auszuführen.
3	J	N X		Eine System-Call-Tabelle enthält die Speicheradressen wichtiger Kernel-Funktionen. Das ermöglicht es Prozessen, diese Funktionen des Kernels direkt aufzurufen.
4	J X	N		Ein Programmable Interrupt Controller (PIC) leitet Interrupts von mehreren Geräten an die CPU weiter; die CPU „fragt“ ihn nach der Interrupt-Nummer.
5	J	N X		Ein CPU-Spezialregister speichert die Adresse des Interrupt-Handlers. Dieser wertet die Interrupt-Nummer aus und ruft dann eine geeignete Funktion zur Bearbeitung auf.
6	J	N X		Mehrere, kurz nacheinander auftretende Interrupts müssen zwingend in serieller Reihenfolge abgearbeitet werden, weil Interrupt-Handler nicht unterbrechbar sind.
7	J	N X		Im User Mode (Ring 3) ist Vollzugriff auf die Hardware möglich, wenn ein Programm diesen vorher über einen System Call anfordert.
8	J X	N		Faults (z. B. Division durch 0, Speicherzugriffsfehler) werden synchron verarbeitet.
9	J X	N		Die Speicherverwaltung muss sicherstellen, dass Prozesse nur ihren eigenen Adressraum sehen. Zugriff auf Betriebssystem-eigene Daten ist (wenn überhaupt) nur über System Calls möglich.
10	J X	N		Klassische PCs verwenden zwei kaskadierte Interrupt-Controller, um die Zahl der unterscheidbaren Interrupt-Quellen (annähernd) zu verdoppeln.

Nr.	Ja	Nein	Weiß nicht	Aussage
11	J X	N		Unterbrechende (präemptive) Scheduler setzen zwingend voraus, dass die CPU Interrupts verarbeiten kann.
12	J	N X		Jeder Hauptspeicherzugriff einer Anwendung, die im User Mode läuft, muss über einen System Call erfolgen. (Nur der Kernel kann direkt auf den Speicher zugreifen.)
13	J X	N		Im Kernel Mode (Ring 0) ist Vollzugriff auf die Hardware möglich.
14	J X	N		Der Zugriff auf eine illegale Adresse löst bei Paging-basierten Betriebssystemen einen Page Fault aus. Das ist ein spezieller Fault, der dann von einem Fault Handler bearbeitet wird, z. B. durch Abbruch des Programms.
15	J X	N		Bei System Calls wird eine System-Call-Nummer übergeben, z. B. über ein Prozessor-Register oder durch Ablage der Nummer auf dem Stack.
16	J	N X		Prozesse können ihre Performance verbessern, indem sie die Interrupts ausschalten: Sie arbeiten dann unterbrechungsfrei.
17	J	N X		An den Interrupt-Eingang der CPU werden mehrere Geräte angeschlossen. Bei einem Interrupt fragt die CPU jedes Gerät, ob es diesen Interrupt erzeugt hat. Aus den Antworten ergibt sich die Interrupt-Nummer.
18	J X	N		Beim Polling fragt das Betriebssystem in einer Schleife solange den Status eines angeschlossenen Geräts ab, bis dieser einen gewünschten Wert annimmt.
19	J X	N		Betriebssystem-Entwickler können entscheiden, ob Interrupt-Handler durch weitere Interrupts unterbrechbar sind (oder Interrupts zwingend in serieller Form bearbeitet werden).
20	J	N X		Prozesse, die auf die Fertigstellung einer I/O-Anfrage warten, rufen in einer Schleife wiederholt einen Interrupt-Handler auf, der ihnen mitteilt, ob die Anfrage fertig bearbeitet wurde.
21	J	N X		Klassische PCs verwenden zwei Interrupt-Controller, die separat an die CPU angeschlossen sind. Damit sind doppelt so viele Interrupt-Quellen unterscheidbar wie beim Einsatz eines einzelnen Interrupt-Controllers.
22	J	N X		Tritt ein Interrupt auf, während ein Prozess läuft (und Interrupts aktiviert sind), wird der Prozess zunächst fortgesetzt. Wenn sein Zeitquantum abgelaufen ist, wird der Interrupt-Handler ausgeführt.
23	J X	N		Den Versuch eines Programms, einen nicht verfügbaren System Call aufzurufen, kann nur das Betriebssystem (nicht die CPU) erkennen.
24	J X	N		Das Deaktivieren der Interrupts ist eine privilegierte Operation, die nur im Kernel-Modus (durch das Betriebssystem) möglich ist.
25	J X	N		Die Gerätekommunikation über Interrupts ist vor allem bei Multi-Tasking-Betriebssystemen sinnvoll: Prozesse werden bei I/O-Anfragen blockiert, bis das Gerät über einen Interrupt signalisiert, dass die Anfrage abgeschlossen ist. In der Zwischenzeit laufen andere Prozesse.
26	J	N X		Bibliotheksfunktionen, die Wrapper für System Calls sind, prüfen, ob der aufrufende Prozess berechtigt ist, die gewünschte Aktion auszuführen. Danach rufen sie den System Call auf, den das Betriebssystem verarbeitet.
27	J X	N		Ein CPU-Spezialregister verweist auf eine Tabelle im Hauptspeicher, die für jeden zulässigen Interrupt die Adresse eines speziellen Interrupt-Handlers für diese Interrupt-Nummer enthält.
28	J	N X		Die Adressen von bis zu 128 System-Call-Handlern speichern Intel-CPUs in einem Array von Spezialregistern. Dieses Array muss vom Betriebssystem bei der Initialisierung gefüllt werden.

Nr.	Ja	Nein	Weiß nicht	Aussage
29	J	N X		Ulix mappt die Interrupt-Nummern 0–7 des ersten PIC sowie 0–7 des zweiten PIC auf die Nummern 16–31. *)
30	J	N X		Ulix trägt mit `lidt` in das CPU-Register *IDTR* die Anfangsadresse der Interrupt-Deskriptor-Tabelle (`struct idt_entry idt[256]`) ein. -> sondern Adresse IDT-Ptr.
31	J	N X		Die Instruktion `sti` schaltet alle Interrupts aus.
32	J X	N		Interrupt-Handler werden bei Ulix im Kernel-Modus ausgeführt.
33	J X	N		Tritt ein Interrupt auf, muss Ulix diesen gegenüber dem PIC bestätigen – bei Interrupts vom zweiten PIC muss es die Bestätigung *an beide PICs* schicken.
34	J	N X		Über die Interrupt-Maske werden alle Interrupts ein- bzw. ausgeschaltet.
35	J	N X		Bevor die CPU in den Interrupt-Handler wechselt, sichert sie automatisch alle CPU-Register auf dem Stack. -> nicht alle!
36	J	N X		Ulix verwendet die Funktion `install_interrupt_handler()`, um die Adresse einer spezifischen Handler-Funktion (für eine bestimmte Interrupt-Nummer) in die Interrupt-Deskriptor-Tabelle einzutragen.
37	J	N X		Interrupt- und Fault-Handler werden von Ulix in separaten Tabellen verwaltet. 1. IDT enthält alle Handler. 2. Es gibt noch die Tabelle interrupt_handlers, aber keine analoge Tabelle fault_handlers

Nr.	Ja	Nein	Weiß nicht	Aussage
38	J X	N		Ulix implementiert System Calls über einen Interrupt-Handler für die Interrupt-Nummer 0x80.
39	J	N X		Ein Programm, das als Prozess ausgeführt wird, führt einen System Call durch, indem es die Syscall-Nummer sowie ggf. Argumente auf den Stack schreibt und dann `int 0x80` ausführt. -> Register, nicht Stack!
40	J	N X		Syscall-Handler sind Teil der Standardbibliothek und mit dem Programm gelinkt. Damit sie korrekt arbeiten können, wechselt das System vor der Abarbeitung mit `int 0x80` in den Kernel-Mode; danach wird die Bearbeitung der Syscall-Handler-Funktion fortgesetzt.
41	J	N X		Ulix unterstützt für jeden System Call bis zu vier Argumente, die in *EAX*, *EBX*, *ECX*, *EDX* gespeichert werden. -> 3 Arg., EAX: Syscall-Nr.
42	J X	N		Ulix verwendet die Funktion `install_interrupt_handler()`, um die Adresse einer spezifischen Handler-Funktion (für eine bestimmte Interrupt-Nummer) in die Ulix-interne Tabelle `interrupt_handlers[]` einzutragen.
43	J X	N		Für jeden System Call, den Ulix unterstützen soll, muss ein Syscall-Handler in die Tabelle `syscall_table[]` eingetragen werden.
44	J	N X		Der Tastatur-Interrupt-Handler liest mit `inportb()` den ASCII-Wert des eingegebenen Zeichens aus. -> Scancode, nicht Zeichen

*) zu 29): Satz sollte eigentlich mit "... auf die Nummern 32-37" enden (und wäre dann korrekt gewesen), so (mit 16-31) ist er falsch . . . . . .