Aufgabenblatt 2

19. April 2025

Aufgabe 2.1

a)

Ein Interrupt wird von einer Software oder Hardware ausgelöst und ermöglicht das Dazwischenrufen von Prozessen. Dabei können Daten an das System weitergegeben werden oder Aktionen auf dem System ausgeführt werden. Polling hingegen ist eine Technik, bei der das System regelmäßig den Status eines Geräts abfragt, um festzustellen, ob eine Aktion erforderlich ist. Das heißt beim Polling übernimmt das System die Kontrolle und Abfrage eines Geräts und beim Interrupt übernimmt das Gerät die Kontrolle.

b)

Es gibt den Hardware Interrupt welcher Asynchron zum Programmcode ausgelöst werden kann, er kann z.B. durch Timer, E/A-Geräte oder auch Drucker ausgelöst werden. Dann gibt es die Software Interrupts, welche Synchron ablaufen und im Prozessor während der Befehlsausführung ausgeführt werden. Durch sie werden Systemaufrufe realisiert.

c)

Nested Interrupts sind essentiell für Echtzeitandwendungen bei denen genaue Reaktionszeit erwartet wird. Ein Nachteil von Nested Interrupts ist die erhöhte Komplexität bei der Verwaltung der Interrupts. Da Interrupts während der Bearbeitung eines anderen Interrupts unterbrochen werden können, muss das System sicherstellen, dass der Kontext des unterbrochenen Interrupts korrekt gespeichert und später wiederhergestellt wird. Dies kann zu einem höheren Speicherbedarf und einer längeren Latenzzeit führen.

d)

Das Betriebssystem muss die Ausführung von Interrupts temporär blockieren können, um kritische Abschnitte zu schützen und Datenkonsistenz sicherzustellen. Wenn ein Interrupt während der Ausführung eines kritischen Abschnitts auftritt, könnte dies dazu führen, dass Datenstrukturen in einem inkonsistenten Zustand verbleiben.

e)

Ein Programm im Userspace hat keinen direkten Zugriff auf Hardware oder privilegierte CPU-Funktionen und kann daher keine Interrupts blockieren.

f)

Wenn ein IRQ ausgelöst wird, während Interrupts temporär deaktiviert sind, wird der IRQ zwischengespeichert und nach Aktivierung der Interrupts verarbeitet.

g)

Ja Interrupts können verloren gehen, wenn sie nicht maskierbar sind oder sie während deaktivierter Interrupts auftreten und nicht gepuffert werden. Auch können fehlerhafte Treiber eine Ursache sein.

Aufgabe 2.2

a)

Systemaufrufe werden durch Software-Interrupts oder Trap-Instruktionen implementiert. Diese Mechanismen erlauben den Übergang vom Userspace zum Kernelspace.

b)

Der Systemaufruf löst eine spezielle CPU-Instruktion aus, die automatisch einen Wechsel in den Kernelmodus bewirkt, dabei übernimmt das Betriebssystem die Kontrolle. Dies ist keine Sicherheitslücke, da nur vordefinierte und kontrollierte Einstiegspunkte im Kernel erreicht werden können, die genau prüfen, was erlaubt ist.

c)

Ein Anwendungsprogrammierer implementiert in der Regel keine Systemaufrufe. Diese werden in Standardbibliotheken des Betriebssystems implementiert und dienen als Schnittstelle zum Kernel.

d)

Außerhalb einiger anderer Systemaufrufe wie für das Managen von Multithreading durch `arch_prctl()`, `set_tid_address`, `set_robust_list` und `rseq` sind die Hauptsystemaufrufe im Code `brk`, welches durch `malloc(msg_len)` ausgelöst wird. Diese Funktion nimmt eine Speicheradresse entgegen und gibt sie im Erfolgsfall zurück.

Der Aufruf `write(1, "Hallo Welt!\n", 12)` gibt den String `Hallo Welt!` an den Standardausgabekanal (1 = `STDOUT`) aus. Dabei werden drei Parameter übergeben: der File-Descriptor, der String und seine Länge. Zurückgegeben wird die Anzahl der geschriebenen Bytes (12).

Abschließend wird ein `exit_group(0)`-Systemaufruf ausgeführt, welcher durch das `return 0;` aus `main()` ausgelöst wird. Im `strace`-Output erscheint hierbei `= ?`, da das Programm terminiert und kein Rückgabewert mehr übergeben wird.