Betriebssysteme

Wintersemester 2019/20

Inhaltsverzeichnis

1 Checkpoint 1

1.1 What is an Operating System?

Keine klare Definition vorhanden. Mögliche Antworten: Es ist eine Schicht zwischen Hard- und Software, welche Programme und Resourcen verwaltet.

1.2 Why did Operating Systems emerge?

Anforderungen an Computer haben sich mit der Zeit verändert. Jobverwaltung und bessere Auslastung wurde immer relevanter. Betriebsysteme wurden entsprechend zur Verwaltung verschiedener Aufgaben auf einem System.

1.3 Which of the following statements is false?

a: Operating System evolution required Hardware changes

b: Hardware evolution required Operating System changes

c: Hardware and OS evolved independently

d: There was strong influence in both directions

Lösung: Antwort c ist falsch.

1.4 Describe three critical early inventions in Operating Systems

mögliche Lösungen sind:

1. Time Sharing
2. (Pipelines)
3. Job Controll (mehrere Programme direkt hintereinander ausführen)
4. Multiprogrammierung (andere Programme laufen, während ein Programm auf IO Events wartet)

[TODO: auf entsprechenden Folien verweisen]

1.5 What is UNIX?

Eine historische Betriebssystemfamilie, die auch heute noch immer aktuell ist.

1.6 What is POSIX?

Eine stark an UNIX angelehnter Quellcodestandard. Beschreibt wie ein System entwickelt werden muss, damit es POSIX Kompatibel ist. (Sich wie ein POSIX System verhält.) Es ist selbst kein Betriebssystem.

1.7 Discuss wether POSIX is still relevant today.

Keine falsche Antwort, da Diskussion. Entscheidend ist die Begründung.

1.8 Describe the relation between the terms Process, Program, Thread and File

Ein Prozess ist laufendes Programm, ein Programm ist eine ausführbare Datei und ein Thread ist Ausführungsstrang eines Prozesses. Ein Prozess kann mehrere Threads haben, welche alle im gleichen Kontext laufen. (Sie teilen sich Speicher etc.)

1.9 What is a shell?

Eine (Eingabe- und Ausgabe-) Umgebung in der eine Eingabe als Befehl interpretiert und entsprechend ausgeführt wird. (Kommandointerpreter)

1.10 Describe briefly what happens when you type 'ls' into a UNIX shell and press enter

Wenn Enter gedrückt wird, wird ein Interrupt erzeugt, welcher vom Betriebssystem abgefangen wird und dann als Eingabe an die Shell gesendet wird. Anschließend wird der Befehl 'ls' von der Shell als Programmaufruf interpretiert woraufhin die Shell sich selbst forkt und im Kind das Programm 'ls' im PATH sucht und ggf. startet. 'ls' nutzt die gleiche Ausgabe wie die Shell und zeigt so alle Dateien. Der Elternprozess wartet darauf, dass der Kindprozess terminiert.

1.10.1 What would have happened if you had typed 'cd' instead?

'cd' wird als aufruf eines Builtin interpretiert, weshalb die Shell in das HOME Verzeichnis des Nutzers wechselt. Es geschieht kein Fork-Exec.

1.11 In Pseudo-Code, write a program that executes 'ls' in a child process and waits in the parent process for the termination of the child process.

```
#include <...>

int main(){
    pid_t pid = fork();
    if(pid == -1){
        printf("error in fork\n");
        return -1;
    }else if(pid == 0){
        // CHILD
        char *const args = {"ls", NULL};
        int ret = execvp("ls", args);
        // ret == -1
        printf("error in exec; errno: %d\n", errno);
        return -1;
    }else{
        // PARENT
        wait(pid);
```

```
            return 0;
        }
    }
```

1.11.1 Which system calls would you use?

wait, fork, exec

1.12 Briefly describe five tasks of an Operating System

Resourcenverwaltung (Storage Managment (Haupt- und Plattenspeicher), Memory Managment, Scheduling (CPU Verwaltung, Prozessormanagment), Device Managment (Geräterverwaltung und Treiber, Interruptbehandlung)), Security- und Usermanagment (auch Prozessschutz voreinander)

1.13 Briefly describe three design goals of an Operating System

Portability, Maintability, Security, Performance, Responsive

1.14 Discuss the reasoning behind the seperation of User- and Kernel Mode. Identify advantages and disadvantages.

Schutz der Programme voneinander (Isolation), Benutzerverwaltung und Berechtigungssystem
Ein Nachteil: extra Performance durch ständige Wechsel über Syscalls

1.15 What is an Interrupt?

Eine Unterbrechung des aktuellen Programmablaufes (meist durch externe Geräte). Syncrone Interrupts entstehen durch Programmfluss (meist durch Exceptions) und Asyncrone Interrupts entstehen durch Hardwareereignisse.

1.16 Describe how the Operating System handles incoming Interrupts.

CPU hält die Ausführung des laufenden Prozesses an, die CPU sichert den Kontext des aktuellen Prozesses und behandelt anschließend den Interrupt entsprechend. (Meist ist der entsprechende Behandlungscode im Treiber) In der Interruptbehandlung wird dem Gerät auch mitgeteilt, dass es jetzt aufhören kann Interrupts zu senden. (Meist gleich als erstes) Anschließend wird das Programm fortgesetzt, indem sein Kontext wiederhergestellt wird. Während der Interruptbehandlung wird Interruptbehandlung weiterer möglicherweise eintretenden Interrupts deaktiviert. Entsprechend darf es während der Interruptbehandlung keine Programmierfehler geben. Interrupts können auch erst später behandelt werden, je nach Priorität.