Betriebssysteme

Wintersemester 2019/20

Inhaltsverzeichnis

1 Checkpoint 1 **3**

1.2 What is an Operating System? . 3

1.3 Why did Operating Systems emerge? 3

1.4 Which of the following statements is false? 3

1.5 Describe three critical early inventions in Operating Systems . . 3

1.6 What is UNIX? 3

1.7 What is POSIX? 3

1.8 Discuss wether POSIX is still relevant today. 4

1.9 Describe the relation between the terms Process, Program, Thread and File 4

1.10 What is a shell? 4

1.11 Describe briefly what happens when you type 'ls' into a UNIX shell and press enter 4

1.11.1 What would have happened if you had typed 'cd' instead? 4

1.12 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. 4

1.12.1 Which system calls would you use? 5

1.13 Briefly describe five tasks of an Operating System 5

1.14 Briefly describe three design goals of an Operating System . . 5

1.15 Discuss the reasoning behind the seperation of User- and Kernel Mode. Identify advantages and disadvantages. 5

1.16 What is an Interrupt? 5

1.17 What can cause an Interrupt? . . 5

1.18 Describe how the Operating System handles incoming Interrupts. 5

1.19 What is the purpose of a System Call? 5

1.20 Describe what happens when at runtime when a program uses the function 'getpid'. 6

1.21 What are Windows "Personalities"? 6

1.22 Describe the anatomy of a Windows Subsystem. 6

1.23 Compare the three types of Subsystem Service Calls. 6

1.24 Name three Operating System components usually located in User- and Kerner Mode (three each) 6

1.25 What is the role of a System Thread in Windows? Name two examples. 7

1.26 What is a Servive / Daemon? . . 7

1.27 Compare the concepts "Microkernel" and "Monolithic Kernel". . . 7

1.27.1 Which one would you use to describe Windows and which one for Linux? . . . 7

1.28 The Windows Kernel is Object Oriented. What is a Handle and what is its role? 7

1.29 Can you share handles between separate Processes? 7

1.30 Describe four types of Windows Kernel Objects. 8

1.31 What is the Basic Input/Output System (BIOS)? 8

2 Checkpoint 2 **9**

2.1 title 9
2.3 What is Preemption? 9
2.4 What new challanges did Preemption introduce when compared to cooperative multiprogramming? 9
2.5 How is Preemption implemented in an operating system kernel? . . 9
2.6 Compare Concurrency and Parallelism. 9
2.7 What is a Critical Section? 9
2.8 What is the value of shared, and why? 9

3 Checkpoint 3 **10**

3.2 What is a Programm? What is a Process? What is a Thread? . . . 10
3.3 What are the roles of the scheduler and the dispatcher? 10
3.4 Compare the long term scheduler to the short term scheduler. . . . 10
3.5 Describe 3 properties related to process and thread control objects. 10
3.6 Describe 3 properties related to the CPU context. 10
3.7 Outline the sequence of a context switch as performed by the dispatcher. 10
3.8 Compare cooperative to preemptive scheduling. 10
3.9 What is a quantum? What effect does the length of a quantum have on the scheduler performance? 10
3.10 Threads can have different states in relation to scheduling. Draw a diagram and Describe the purpose of the states and their transitions. 11
3.11 When are scheduling decisions made? 11
3.12 What are the optimiztion criteria of a scheduler? 11
3.13 Draw a FIFO Gantt Chart for the given taskset. Assume a context switch takes no time. 11
3.14 Draw a Round Robin Gantt chart for the given taskset. Assume a context switch takes not time, use quantum of length 3 12

1 Checkpoint 1

1.2 What is an Operating System?

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

1.3 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.

1.4 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.

1.5 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

1.6 What is UNIX?

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

1.7 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.

1.8 Discuss wether POSIX is still relevant today.

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

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

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

1.10 What is a shell?

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

1.11 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.

1.11.1 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.

1.12 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;
        }
    }
```

1.12.1 Which system calls would you use?

wait, fork, exec

1.13 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)

1.14 Briefly describe three design goals of an Operating System

Portability, Maintability, Security, Performance, Responsive

1.15 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

1.16 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.

1.17 What can cause an Interrupt?

Ein Interrupt kann zum Beispiel ausgelöst werden, wenn ein Prozess auf IO Zugriff warten muss.

1.18 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.

1.19 What is the purpose of a System Call?

System Calls werden verwendet um von dem User Mode aktionen im Kernel Mode zu initialisieren.

1.20 Describe what happens when at runtime when a program uses the function 'getpid'.

Wir bekomme die PID(werden benötigt um Prozessen von einander zu entscheiden) des aktuellen Prozesses. Funktion ist in der User Mode Bibliothek beschrieben. Funktion wird aufgerufen und schreibt in ein Register die Syscall Nummer. Cpu wechselt in den Kernel Mode und die Routine für Systemaufrufe laden. Schaut die Syscall Nummer im Register nach und führt dann die entsprechende Funktion hier getpid() aus. Systemaufruf ist ein Synchroner Interrupt.
Funktion → User Mode Bibliothek → Syscall, Softwareinterrupt → Interraptionsroutine bestimmt Funktion an Hand der Nummer im Register → Funktion ausführen und Ergebniss nach Oben schleifen

1.21 What are Windows "Personalities"?

a: Independent operating modes of the CPU

b: Distinguished Engineers at the Microsoft Campus in Redmond

c: Seperate classes of applications, and their corresponding subsystems

d: Groups of Users in the System with different Privileges

Antwort C

1.22 Describe the anatomy of a Windows Subsystem.

Programme benutzen Subsystems an der Stelle von APIs. Die Subsysteme sind dokumentieren und greifen auf undokumentierte Windows System Service Calls zu.

1.23 Compare the three types of Subsystem Service Calls.

- vollständig im User Mode implementiert → Geometriefkt. PtInRect() IsRectEmpty()
- es werden ein oder mehrere Systemcalls für die Ausführung benötigt → ReadFile() WriteFile(); erstellt ind Subsystem Library
- benötigt die Umgebung des Subsystem Process → CreateFile(), durch IPC

1.24 Name three Operating System components usually located in User- and Kerner Mode (three each)

- User Mode
  - Commandointerpreter (Shell)
  - Subsysteme
  - Services, Dienste
- Kernel Mode
  - Grafikengine
  - Hardware Abstraction, Treiber, ISR
  - Kernel

- Memorymanagement
- Management...

1.25 What is the role of a System Thread in Windows? Name two examples.

Thread ist ein Handlungsstrang eines Prozesses im User Mode.
Ein Systemthread ist eine nebenläufige Aktivität im Kernel Mode, welche eine spezielle Aufgabe erfüllt. Ein Systemthread wird durch das System nicht als Prozess abgebildet. Ein Systemthread hat priviligierten Zugriff auf das System.

Beispiele: Zero Page Thread (Nullt Speicher), eigene Threads von Treibern, Balance Set Manager (greift bei Speicherknappheit ein und verwaltet Prozessorspeicher), generell alle nebenläufigen Aufgaben im Kernel

1.26 What is a Servive / Daemon?

Hintergrundprozess im System, Kind von init, hat keine Ein- oder Ausgabe, vollführt Systemrelevante Aufgaben, üblicherweise ist er nicht-priviligiert

1.27 Compare the concepts "Microkernel" and "Monolithic Kernel".

Microkern: so viel Kernel Funktionalität wie möglich in den User Mode verlegen. Der Kernel enthält nur noch die wirklich relevanten Dinge. (Scheduling, Speicherverwaltung, Dispatch, Interprozesskommunikation) Meist aus Sicherheitsgründen. Entsprechend sind alle Treiber und die Kernel-Objektverwaltung im User Mode und werden isoliert. Zusammenfassung: kleiner Kern mit wenig Funktionen, vielen Diensten

Monolithic Kernel: mächtiger Kern mit vielen Funktionen im Kernel Mode, wenig Diensten

1.27.1 Which one would you use to describe Windows and which one for Linux?

Linux: Monolith Windows: (hybrider Kern,) hauptsächlich Microkern

1.28 The Windows Kernel is Object Oriented. What is a Handle and what is its role?

Ein Handle ist ein Verweis auf ein existierendes Kernel Obejekt. Ein Handle gilt nur für jeweils einen Prozess. (Jeder Prozess hat eine eigene Handle Tabelle.) Ein Handle kann genutzt werden, um Systemfunktionen Kernel Objekten zu übergeben ohne das der Kernel diese Objekte "preisgeben" muss.

1.29 Can you share handles between separate Processes?

Nicht direkt. Aber ich kann in die Handle Tabelle eines anderen Prozesses eine weiteres Handle hinzufügen, welches auf das gleiche Kernel Objekt verweist. (Mit duplicateHandle oder so.)

1.30 Describe four types of Windows Kernel Objects.

File Object (einer geöffneten Datei), Port Object (kann genutzt werden um Nachrichten zwischen Prozessen zu verschicken), Thread Object, Process Object, Symbolic Link, Object directory (kann andere Objekte speichern um Hierarchie zu realisieren), Event Object, Semaphore Object, ... (siehe Unit 1, Vorlesung 3, Folie 30ff.)

1.31 What is the Basic Input/Output System (BIOS)?

Eine Software welches direkt auf der Hardware (der ROM (Read Only Memory)) liegt, welches das aktuell laufende Motherboard booten soll.

2 Checkpoint 2

2.1 title

=======

2.3 What is Preemption?

In einem System in dem Prozesse in Zeitscheiben unterteilt werden, ist dies die faire Aufteilung dieser Zeitscheiben auf die Prozesse, durch das Unterbrechen des aktuellen Prozesses vom Betriebssystem. Jeder Prozess erhält ungefähr gleich viel Rechenzeit.

2.4 What new challanges did Preemption introduce when compared to cooperative multiprogramming?

Programme können an beliebigen Stellen unterbrochen werden. Dies wird zu einem Problem, falls eine Resource zwischen verschiedenen Programmen geteielt wird und beide Programme darauf zugreifen.

2.5 How is Preemption implemented in an operating system kernel?

Meist durch einen Timer Interrupt. (Eine Zeitscheibe endet, wenn die Clock ein Signal gibt.) Der Scheduler wählt ein neues Programm aus, welches Rechenzeit erhalten soll und der Dispatcher tauscht den aktiven Kontext aus, damit das neue Programm rechnen kann.

2.6 Compare Concurrency and Parallelism.

Concurrency: keine echte Gleichzeitigkeit, sondern eher ein konstantes abwechseln (immer nur aktiver Thread)
Parallelism: zwei Kerne, die echt Gleichzeitig rechnen

2.7 What is a Critical Section?

Ein Codesegment, in dem auf eine geteilte Resource zugegriffen wird.

2.8 What is the value of shared, and why?

In einer Nebenläufigen Ausführung von func_a und func_b:

```
int shared = 0;
void func_a(void)
{
    shared++;
}
```

```
void func_b(void)
{
    shared--;
}
```

Antwort: -1, 0 oder 1; je nachdem wie die Unterbrechung der Programme erfolgen, da das keine atomaren Anweisungen sind und wir uns somit temporäre Werte in Registern merken müssen, welche durch die Unterbrechung invalide werden.

3 Checkpoint 3

3.2 What is a Programm? What is a Process? What is a Thread?

Ein Programm ist eine ausführbare Datei.
Ein Prozess ist ein laufendes Programm.
Ein Thread ist ein Ausführungsstrang eines Programms.

3.3 What are the roles of the scheduler and the dispatcher?

Der Scheduler hat die Aufgabe den laufenden Prozessen ihre CPU Zeit zuzuweisen.
Der Dispatcher gibt die Kontrolle zum dem nächsten aktiven Thread(wiederherstellen des CPU Kontextes, vorbereiten des Adressbereiches für den nächsten Prozess).

3.4 Compare the long term scheduler to the short term scheduler.

Long-term scheduler(Steuert Grad der Multiprogrammierung, kann Prozesse schlafen legen, optional)
Short-term scheduler(Sucht zu jedem Quantum einen neuen Thread aus, Scheduler über den wir immer reden)

3.5 Describe 3 properties related to process and thread control objects.

- Prozess: Speicherbereich, Pfad der ausführbaren Datei
- Thread: CPU Kontext, Schedulingzustand
- Beides: Nicewert(Prioritäten), ID

3.6 Describe 3 properties related to the CPU context.

Prozessorstatuswort, Stackregister, Datenregister, Programmcounter

3.7 Outline the sequence of a context switch as performed by the dispatcher.

Aktuellen Thread unterbrechen → Aktuellen Zustand sichern im Hauptspeicher → Daten vom anderen Thread zurückschreiben → anderen Thread ausführen

3.8 Compare cooperative to preemptive scheduling.

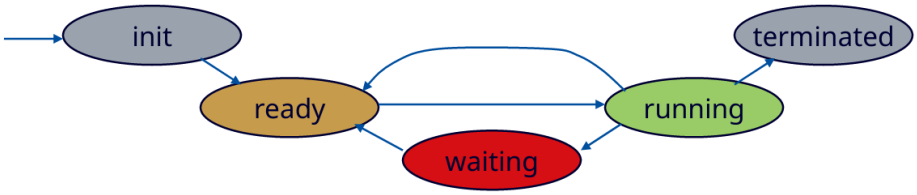
cooperative freiwillige Abgabe der CPU
preemptive Unterbrechung, wenn Quantum abgelaufen

3.9 What is a quantum? What effect does the length of a quantum have on the scheduler performance?

Ein Quantum ist eine feste Zeitscheibe, welche angibt wie lange ein Prozess rechnen darf, bis er unterbrochen wird. Längeres Quantum sorgt für eine höhere Performanz. Kürzeres Quantum ist responsiver.

3.10 Threads can have different states in relation to scheduling. Draw a diagram and Describe the purpose of the states and their transitions.

- **init**: The thread is being created
- **ready**: The thread is waiting to be assigned to a CPU
- **running**: The threads instructions are being executed
- **waiting**: The thread is waiting for some event to occur
- **terminated**: The thread has finished execution



3.11 When are scheduling decisions made?

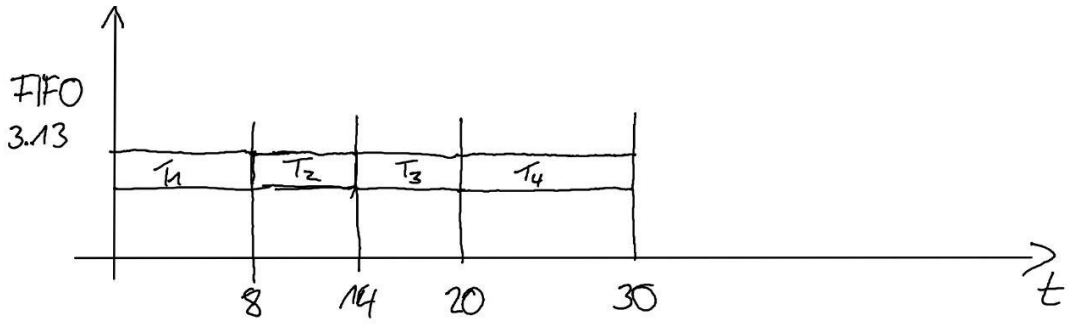
Am Ende von jedem Quantum. Welcher Thread als nächstes in **running** darf. Immer wenn ein neuer thread im system ankommt.

3.12 What are the optimiztion criteria of a scheduler?

- Resposivität
- Durchsatz
- Wartezeit
- Turn Arount Time(Zeit vom ersten Ausführen eines Threads bis zum letzten Ausführen)
- Auslast der CPU

3.13 Draw a FIFO Gantt Chart for the given taskset. Assume a context switch takes no time.

Thread	Arrival Time	Burst Time
T_1	0	8
T_2	2	6
T_3	2	6
T_4	4	10



3.14 Draw a Round Robin Gantt chart for the given taskset. Assume a context switch takes not time, use quantum of length 3

Thread	Arrival Time	Burst Time
T_1	0	8
T_2	2	6
T_3	2	6
T_4	4 + 0	10