TI2-Fragestunde
14.02.2024

Ute Bormann

Fachgespräche

- Regulär in Präsenz, MZH 6140
  (bei Bedarf auch online möglich)
- 30 min pro Gruppe (+ i.d.R. Tutor)
- Wesentliche Grundlage: Übungszettel
  + Fragenlisten (+ kleine Aufgaben) in Folien
- max. Punkte aus Übungen = 100 % ⇒ Vornote
  (nur bei mind. 50% der Punkte)
- Vornote (40%) + Fachgespräch (60%) ⇒ Endnote
- ggf. Wiederholung (einzeln, 15 min)

1. Überblick Betriebssysteme – Fragen

1. Welche zwei Hauptaufgaben hat ein Betriebssystem?
2. Was ist ein *Prozess*?
3. Welche Aufgabe hat ein *Kommando-Interpreter* (z.B. in Unix die Shell)?

2. Das Unix-Dateisystem aus Nutzersicht – Fragen

1. Wie ist ein Unix-Dateisystem strukturiert?
2. Wie können Dateien darin (eindeutig) aufgefunden werden?
3. Was ist ein *Hard Link*? Was ist ein *Symbolic Link*?
4. Ist das Unix-Dateisystem wirklich ein Baum? Begründung.
5. Welche Zugriffsrechte kann man auf eine Unix-Datei haben?
6. Wer darf was bei folgenden Zugriffsrechten auf eine normale Datei?
   - `chmod 640 bla`
   - `chmod 511 blub`
7. Welche Vorteile bietet es, auf Geräte in Unix wie auf Dateien zuzugreifen?
8. Was versteht man unter *Ein-/Ausgabeumlenkung*?

3. Unix-Prozesse aus Nutzersicht – Fragen

1. `bla` sei ein im Pfad ausführbares Programm. Was ist der Unterschied zwischen dem Aufruf `bla` und dem Aufruf `bla &` in der Shell? Welche Auswirkungen hat dies, wenn `bla` von *Standard Input* liest bzw. auf *Standard Output* schreibt?
2. Wie kann ein Nutzer herausfinden, welche Prozesse in einem Unix-System gerade existieren?
3. Was ist eine *Pipe*?
4. Wofür könnte man das folgende Shell-Kommando verwenden?

   ```
   mv *.pdf bla
   ```

5. Wie macht man ein soeben editiertes Shell-File ausführbar?

4. Vom Quellcode zum Programm in Ausführung – Fragen

1. In welche Bereiche (*Segmente*) ist der *(virtuelle) Adressraum* eines Programms in Ausführung in Unix unterteilt, und welche Eigenschaften kennzeichnen sie?
2. Wozu wird der *Stack* verwendet?
3. Welchem Zweck dienen Bibliotheken (*Libraries*)?
4. Welche Aufgabe erfüllt ein *Linker*?
5. Welchen Vorteil hat es, Bibliotheken mit *Position Independent Code* zu versehen?
6. Wozu wird beim Assemblieren eine *Symboltabelle* angelegt?

5. Traps vs. Interrupts vs. Signale – Fragen

1. Worin unterscheidet sich der *Kernel-Mode* vom *User-Mode* (in Unix)? Warum wird diese Unterscheidung getroffen?

2. Was passiert in etwa bei einem *Systemaufruf*? (Reihenfolge der Arbeitsschritte.)

3. Was ist ein *Trap*? Nenne Beispiele.

4. Was ist ein *Interrupt*? Nenne Beispiele für mögliche Interrupt-Quellen. Warum werden sie unterschiedlich priorisiert? Wie wird ein Interrupt in etwa behandelt?

5. Inwiefern unterscheiden sich Traps von Interrupts?

6. In welchen Kontexten finden Betriebssystemaktivitäten statt?

7. Aus welchen Komponenten besteht ein typisches Betriebssystem?

8. Was ist ein *Signal*? Nenne Beispiele für mögliche Signalquellen. Wie kann ein Prozess auf ein Signal reagieren?

6. Prozessverwaltung 1 – Fragen

1. Nenne verschiedene Gründe für eine Prozessumschaltung.
2. Nenne einige Randbedingungen, auf die man beim Entwurf eines *Schedulers* achten sollte. Wie sollten rechenintensive bzw. Ein-/Ausgabe-intensive Prozesse dabei behandelt werden?
3. Wie könnte man mit Hilfe eines *Round-Robin-Schedulers* Prozessprioritäten „simulieren“?
4. Warum bestehen die *Sleep-* und die *Run-Queue* in Unix nicht aus jeweils einer einzigen Warteschlange? Wie sind sie stattdessen organisiert?
5. Beschreibe kurz einige Zustände, in denen sich ein (Unix-)Prozess befinden kann.
6. Warum werden die Zustandsinformationen eines Unix-Prozesses teilweise in der *Proc-Struktur* und teilweise in der *User-Struktur* abgelegt? Nenne jeweils zwei charakteristische Beispiele für Angaben darin.

7. Prozessverwaltung 2 – Fragen

1. Skizziere kurz die Prozesserzeugung in Unix. Welche Rolle spielen die Systemaufrufe `fork()` und `exec()` dabei?

2. Wie erfährt ein Unix-Prozess, ob ein Kindprozess terminiert ist?

3. Kann ein Kindprozess weiterlaufen, wenn sein Vaterprozess bereits terminiert ist?

4. Wozu gibt es in Unix den Prozesszustand SZOMB („Zombie“)?

5. Beschreibe ein typisches Nebenläufigkeitsproblem.

6. Was versteht man in einem Einprozessor-Unix unter *Nicht-Präemption*? Welche Auswirkung hat dies auf mögliche Nebenläufigkeitsprobleme?

8. Speicherverwaltung 1 – Fragen

1. Wie verändert sich die Größe der verschiedenen Segmente des Prozess-Adressraums beim Programmablauf?
2. Welche Vor- und Nachteile hat der *First-Fit*- bzw. der *Best-Fit*-Algorithmus zur Speicherverwaltung?
3. Wie arbeitet der *Buddy-Algorithmus* in etwa?
4. Wo tritt *interne Fragmentierung*, wo *externe Fragmentierung* auf? Was ist das?
5. Wozu bieten Systeme eine *Speicherhierarchie* an? Welche Beobachtung über den Speicherzugriff realer Programme liegt dem zugrunde? Welche verschiedenen Arten von Speicher werden typischerweise bereitgestellt?
6. Warum ist es in der Regel nicht sinnvoll, den Adressraum eines Prozesses in einem Stück im Hauptspeicher abzulegen?
7. Was versteht man unter *Paging*, was unter *Segmentierung*?
8. Aus welchen Teilen besteht eine *virtuelle Adresse* zumeist? Wie ermittelt sich daraus die entsprechende Hauptspeicheradresse, d.h. wie läuft die Adressverwaltung in etwa ab?
9. Wie können mehrere Prozesse mit Hilfe virtueller Adressierung auf dieselben Programmstücke (oder auch Datenbereiche) zugreifen?

9. Speicherverwaltung 2 – Fragen

1. Warum ist ein perfekter Algorithmus zur Verdrängung von Pages aus dem Hauptspeicher nicht realisierbar?

2. Wie arbeiten die folgenden Algorithmen in etwa:

   a) FIFO (First-In-First-Out),
   b) LFU (Least-Frequently-Used),
   c) LRU (Least-Recently-Used)?

3. In welche dieser Kategorien kann man NRU (Not-Recently-Used) einordnen?

4. Wie arbeitet der *Clock-Hand-Algorithmus*?

5. Was passiert, wenn die Umlaufzeit des Zeigers beim Clock-Hand-Algorithmus zu groß bzw. zu klein gewählt wird? Wie kann ein zweiter Zeiger den Algorithmus verbessern?

6. Was ist *Swapping*? Warum wenden auch Paging-Systeme dieses Verfahren an bzw. unter welcher Bedingung?

7. Wie kann man die Vorteile von *Paging* und *Segmentierung* kombinieren?

8. Wozu bzw. wo wird bei der Speicherverwaltung häufig ein *Assoziativspeicher* eingesetzt?

10. Dateiverwaltung 1 – Fragen

1. Aus welchen grundlegenden Komponenten besteht ein Dateisystem?

2. Wie sieht die Struktur des Unix-V7-Dateisystems auf der Platte in etwa aus? Warum erfolgt die Verwaltung der Freispeicherliste über Indirekt-Blöcke?

3. Welche Angaben enthält ein *Inode*? Welche Angaben enthält eine *Verzeichnis-Datei*?

4. Beschreibe kurz die Zugriffsoperationen `open()`, `close()`, `lseek()`, `read()` und `write()` auf ein Unix-Filesystem. Welche Rolle spielt der *Filedeskriptor* dabei?

5. Was geschieht durch einen `mount()`-Systemaufruf in etwa?

11. Dateiverwaltung 2 – Fragen

1. Wozu werden die folgenden Kern-internen Datenstrukturen verwendet?
   a) Deskriptor-Tabelle
   b) File-Tabelle
   c) Inode-Tabelle

2. Welche Aufgaben hat der *Buffer Cache* in Unix?

3. Welche Vorteile bietet es, Dateien mit dem Unix-Systemaufruf `mmap()` in den virtuellen Adressraum eines Prozesses abzubilden?

12. Geräteverwaltung 1 / Dateiverwaltung 3 – Fragen

1. Wie ist eine Platte intern organisiert?
2. Wie wirkt sich dies auf den Informationszugriff aus?
3. Wie geht das Unix *Fast File System* damit um?
4. Was versteht man beim Zugriff auf Plattenblöcke unter dem *Fahrstuhlalgorithmus*?
5. Welche wesentlichen Eigenschaften hat eine SSD *(Solid State Disk)*?
6. Nenne Beispiele für potentielle Inkonsistenzen in einem Dateisystem. Wie können sie entstehen? Wie kann man sie erkennen/beheben?
7. Welche wesentlichen Eigenschaften hat das *ZFS-Filesystem*?

13. Geräteverwaltung 2 / Booten – Fragen

1. Was ein *Geräte-Controller*? Welche Aufgaben hat er?
2. Welche Vorteile bietet eine vereinheitlichte Betriebssystemschnittstelle zum Zugriff auf Geräte? Wie sieht sie in Unix in etwa aus?
3. Was ist ein *Gerätetreiber? Welche Aufgaben hat er?*
4. Warum erfolgt der Zugriff auf Geräte häufig über Warteschlangen?
5. Worin unterscheidet sich *DMA (Direct Memory Access)* von *Programmed I/O*?
6. Warum werden Terminal-Treiber in Unix parametrisiert? Nenne typische Parameter.
7. Nenne einige wesentliche Aufgaben beim Booten eines Unix-Systems.

14. Nebenläufigkeit – Fragen

1. Skizziere kurz einige Probleme des *nebenläufigen* Zugriffs auf Betriebsmittel.
2. Grenze die Begriffe *Nebenläufigkeit*, *Quasi-Parallelität* und *Parallelität* voneinander ab.
3. Was verstehen wir unter *Nichtdeterminismus*?
4. Welche Nebenläufigkeitseigenschaften bzw. -probleme werden durch die vier folgenden „klassischen“ Szenarien ausgedrückt:
   a) Kritischer Abschnitt *(Critical Section)*,
   b) Leser/Schreiber *(Reader/Writer)*,
   c) Erzeuger/Verbraucher *(Producer/Consumer)*,
   d) Speisende Philosophen *(Dining Philosophers)*?
5. Was versteht man unter *einseitiger* bzw. *mehrseitiger* Synchronisation? Gib jeweils ein Anwendungsbeispiel an.
6. Auf welche verschiedene Arten kann man *Verklemmungen* angehen? Wie arbeitet der Bankiersalgorithmus ?

15. Multithreading – Fragen

1. Was ist ein *Thread*? Skizziere ein sinnvolles Anwendungsbeispiel für die Verwendung mehrerer Threads innerhalb eines Prozesses.

2. Grenze den Thread-Begriff gegen den Unix-Prozess-Begriff ab (Adressraum, Zustandsinformationen, etc.).

3. Die Routinen `pthread_create()`, `pthread_join()`, `pthread_exit()` realisieren die Erzeugung und Termination von Threads in der Unix-Multithreading-Umgebung. Vergleiche ihre Funktionalität mit den Systemaufrufen zur Erzeugung und Termination von Prozessen (`wait()`, `fork()` und `exit()`). Warum arbeitet `pthread_create()` deutlich anders als `fork()`?

16. Realisierung von lock()/unlock() – Fragen

1. Wie kann man den gegenseitigen Ausschluss gewährleisten? Warum ist ein Unterbrechungsausschluss dabei nicht immer das geeignete Mittel?
2. Nach welchen Kriterien wird die Korrektheit bzw. Güte von Locking-Algorithmen bewertet?
3. Warum sollte man die Bewertung von Locking-Algorithmen auf der Grundlage von unteilbaren Operationen durchführen?
4. Skizziere einen einfachen Locking-Algorithmus.

17. Locks vs. Ereignisvariablen – Fragen

1. Warum kann man das Leser/Schreiber-Problem nicht mit einem einzigen Lock lösen?
2. Wie kann man eine einseitige Synchronisation mit Hilfe von `wait()` und `signal()` vornehmen? Wie kann man diese Primitiven in etwa auf `lock()` und `unlock()` abbilden?
3. Grenze die Begriffe *aktives* und *blockierendes* Warten gegeneinander ab.
4. In einer Unix-Multiprozessorumgebung können mehrere Prozesse nebenläufig `sleep()`/`wakeup()` aufrufen. Warum ist dies ein kritischer Abschnitt? Warum kann man ihn nicht einfach dadurch schützen, dass man den Aufruf von `sleep()`/`wakeup()` von einem *Spinlock* umgibt? Was wird man stattdessen tun?

18. Semaphore – Fragen

1. Welche zusätzlichen Eigenschaften zeichnen *Semaphore* gegenüber blockierenden Locks aus?
2. Wie wird eine einseitige bzw. eine mehrseitige Synchronisation durch Semaphore ausgedrückt?
3. Wie können Semaphore zur Lösung des Problems der *speisenden Philosophen* eingesetzt werden?
4. In welches Problem wird eine allzu „einfache“ Semaphor-Implementierung laufen?
5. Welche Probleme gibt es mit „fairen Semaphoren“?
   Was sind „Konvois“, und was sind „donnernde Herden“?

19. Monitore – Fragen

1. Was ist ein *Monitor* (als Synchronisationsverfahren)?
2. Unter welchen Bedingungen wird ein Monitor betreten bzw. wieder verlassen?
3. Mit Hilfe welcher Synchronisationsmechanismen können Monitore modelliert werden?

20. Petri-Netze – Fragen

1. Aus welchen Komponenten besteht ein Petri-Netz (mit Marken)? Was kann man damit beschreiben?

2. Was kennzeichnet lebendige bzw. todesgefährdete Petri-Netze?

3. Wie kann man durch ein Petri-Netz typische Synchronisationsvorschriften ausdrücken?
   a) Sequenz
   b) Beschränkte Nebenläufigkeit
   c) Alternative
   d) Unabhängigkeit

4. Wie kann man den Schutz eines *kritischen Abschnitts* bzw. ein *Erzeuger-/Verbraucher*-Szenario mit Hilfe eines Petri-Netzes modellieren?

21. Nachrichtenaustausch – Fragen

1. Was versteht man unter *synchronem* bzw. *asynchronem* Nachrichtenaustausch? Inwiefern sind diese beiden Kommunikationsformen aufeinander abbildbar?

2. Wie kann man die Synchronisationseigenschaften von synchronem bzw. asynchronem Nachrichtenaustausch mit Semaphoren modellieren?

3. Wozu verwendet man *Kanäle* bzw. *Ports*? Was ist das?

4. Was ist ein *guarded command*? Warum kann die Verwendung eines solchen Konzepts gerade in Zusammenhang mit Nachrichtenaustauschvorgängen interessant sein?

5. Welche Nachteile hat die CSP-Lösung des „Sieb des Eratosthenes“?

22. Interprozesskommunikation in Unix – Fragen

1. Was ist *STM (Software Transactional Memory)*?
2. Worin unterscheiden sich die Eigenschaften der folgenden Unix-Mechanismen zur Interprozesskommunikation:
   a) *Pipes*,
   b) *Named Pipes*,
   c) *Sockets*?
3. Wie lässt sich der Zugriff auf Sockets in die generische Systemaufrufschnittstelle zum Zugriff auf Dateien einordnen?
4. Wie können Sockets adressiert werden?
5. Wie arbeitet der *Korridor-Algorithmus* zum Überwachen mehrerer Eingabequellen in etwa?

23. Rechnernetze 1 – Fragen

1. Skizziere kurz einige typische Kommunikationsprobleme und je eine mögliche Lösung..
2. Skizziere einige Eigenschaften typischer *Netztopologien*.
3. Welche besondere Bedeutung kommt IP zu?
4. Was ist ein *Remote Procedure Call (RPC)*, und welche Parameter wird man typischerweise dabei übergeben?
5. Was ist ein *(Kommunikations-)Protokoll*?

24. Rechnernetze 2 – Fragen

1. Wie kann eine Nachricht im Grundsatz über eine Hierarchie von Protokollen versendet werden?
2. Aus welchen Teilen besteht eine IP-Adresse?
3. Worin unterscheiden sich IPv4 und IPv6?
4. Wozu enthalten IP-Pakete ein TTL-Feld?
5. Welche wesentlichen Aufgaben hat TCP?
6. Was kann man mit Hilfe von DNS ermitteln?
7. Wozu benötigt man URLs?
8. Welchen grundsätzlichen Aufbau hat HTTP?

25. Informationssicherheit – Fragen

1. Nenne einige absichtliche und unabsichtliche Angriffe auf Hardware, Software und/oder Daten.
2. Welche grundsätzlichen *Sicherheitsziele* kann man unterscheiden?
3. Was ist eine *Sicherheitspolitik*?
4. Auf welche verschiedenen Arten kann sich ein Benutzer authentifizieren?
5. Welche Komponenten enthält eine Zugriffskontrollmatrix? Wie ordnen sich die Dateizugriffsrechte in Unix in dieses Schema ein?
6. Charakterisiere *symmetrische* und *asymmetrische* Verschlüsselungsverfahren. Wie können sie zur Realisierung einer Vertraulichkeit eingesetzt werden? Warum werden häufig Mischformen eingesetzt?
7. Was ist Diffie-Hellman?
8. Was sind Zertifikate?
9. Wie kann eine *digitale Unterschrift* erzeugt werden?