6. Aufgabenblatt

11. Juni 2000

Aufgabe 1

Was passiert, wenn eine Datei geöffnet wird?

1. Mit welchen Sprachmitteln wird in C++ (in C) eine Datei geöffnet? Erlären Sie die Wirkung dieser Sprachkonstrukte.
2. Welchen Systemaufruf stellt das Betriebssystem Unix zur Verfügung, um eine Datei zu öffnen? Welche Wirkung hat ein solcher Aufruf?
3. Was passiert auf der Platte, wenn eine Datei geöffnet wird: Wodurch unterscheidet sich dort eine göffnete von einer nicht geöffneten Datei?
4. Warum werden Dateien vor Lese–/Schreibzugriffen geöffnet.
5. Ist ein Betriebssystem denkbar, bei dem Dateien vor einem Zugriff nicht geöffnet werden müssen? Kann auf einem solchen System ein C++/C–Programm ausgeführt werden, oder ist dies prinzipiell unmöglich?

Aufgabe 2

Zugriffsrechte sind konzeptionell als Matrix A darstellbar: Jedem Benutzer u und jeder Datei d ist eine Menge von Zugriffsrechten $R = A(u, d)$ zugeordnet. Unterschiedliche Systeme speichern diese Informationen in unterschiedlicher Art und berechnen dann auch auf unterschiedliche Art die einem Benutzer und einer Datei zugeordneten Rechte.

1. Aus welchen Elementen kann R in Unix bestehen: Welche Rechte auf einer Datei gibt es also?
2. Wieviele Elemente enthält die in Unix einem Benutzer u und einer Datei d zugeordnete Menge an Rechten: ist $R = A(u, d)$ genau einelementig, kann R leer sein, beliebig viele Elemente enthalten, ...?
3. Wie wird A in Unix konkret gespeichert und wie wird $R = A(u, d)$ berechnet (festgestellt)?
4. Wie wird in einem System mit ACLs die Matrix A gespeichert?
5. Erläutern Sie die Vor– und Nachteile einer ACL.
6. Wie wird in Unix (ohne ACLs) einer Gruppe von 10 Benutzern Schreibrecht an 5 Dateien gegeben?

7. Kann weiteren 100 Benutzern Leserecht an den gleichen 5 Dateien gegeben werden, ohne dass dies die Rechte der ersten 10 Benutzer beeinträchtigt oder anderen Benutzern irgendwelche Rechte an diesen Dateien gegeben werden? Wie ist es wenn ACLs zur Verfügung stehen?

8. Programme werden in Unix mit der Identität ihres "Aktivators" ausgeführt. Wenn Benutzer u also ein Programm startet, dann hat dieses die gleichen Rechte wie u. Es ist jedoch möglich Programme zu schreiben die während ihrer Ausführung die Identität wechseln. Ein Programm kann auch stets problemlos die Identität seines ursprünglichen Aktivators feststellen. Das gleiche gilt für die Gruppenzugehörigkeit.Erläutern Sie in wie weit mit diesem Mechanismus eine ACL simuliert werden kann?

Aufgabe 3

1. Was passiert, wenn eine Platte *low–level* oder *high–level* formatiert wird. Werden dabei Informationen auf der Platte abgelegt? Wenn ja: welche und für wen sind sie bestimmt?

2. Zur Erhöhung der Datensicherheit von Platten werden ECCs abgespeichert. Was ist das, wo wird es abgespeichert, wer verarbeitet es wann?

3. Ist ein Sektor identisch mit einem Block, ist er größer oder kleiner, kann ein Sektor mehrere Blöcke enthalten?

4. Was ist eine Partition?

5. Was ist eine Swap–Partition?

6. Was ist ein Dateisystem?

7. Kann ein Dateisystem größer sein als: eine Partition, eine Platte, ein Verzeichnis?

8. Kann ein Verzeichnis größer sein als: eine Partition, eine Platte, ein Dateisystem?

9. Was bedeutet FAT?

10. Was ist ein *inode*?

11. Daten auf Platten werden durch "Redundanz" vor Datenverlust geschützt. Was bedeutet der Begriff "Redundanz"? Welche Redundanz steckt in der Hardware (Platte/Controller) welche Redundanz liefert das Betriebssystem? Erläutern Sie mit welchen Mechanismen die Platte selbst sowie ein FAT– und ein ext2–Dateisystem Fehler reparieren (oder dies zumindest versuchen).

12. Im Dateisystemen von Unix gibt es keine "Laufwerke" wie in DOS oder Windows. Was ist ein "Laufwerk" und wie wird die entsprechende Funktionalität in Unix realisiert – falls sie dort überhaupt existiert?

13. Durch welchen Mechanismus wird in Unix die Existenz verschiedener Dateisysteme "transparent" gemacht?

14. Erläutern Sie die in der Informatik üblichen Bedeutungen der Begriffe "transparent" und "virtuell".

Aufgabe 4

Jedes Betriebssystem definiert eine oder mehrere Satztypen, als die – für Anwendungen – les- und schreibbare Einheiten auf einer Datei.

1. Welchen Satztyp haben Unix und Windows–NT?
2. Der Systemaufruf `write` hat folgende Aufrufschnittstelle:

   ```
   #include <unistd.h>
   ssize_t write(int fd, const char *buf, size_t count);
   ```

   Stecken hierin irgendwelche Inforamtionen über den Satztyp?
3. Hat C++/C ein eigenes Satzkonzept? Gibt es also in C++/C (der Standardbibliothek) unabhängig vom Betriebssystem eine feste Informationseinheit auf der alle Datentransfers von und zu Dateien aufbauen? Wenn ja: welche ist das? Wenn nein: wie werden Satztypen in C++/C definiert?
4. In welcher Beziehung stehen Blockgröße und Satztyp: sind sie identisch, ist der eine immer kleiner als der andere, größer, kein Bezug, ...?

Aufgabe 5

Die Programmiersprache Pascal hat ein Dateikonzept, bei dem jedes Programm nach Belieben Satztypen definieren kann. Beispiel:

```
TYPE R = RECORD ....;   Typdefinition: R ist ein Typ
TYPE F = FILE OF R;     Typdefinition: F ist der Typ der Dateien mit Satztyp R
f : F;                  f ist eine Datei mit Satztyp R
r : R;                  r ist eine Variable vom Typ R
...
read(f,r);              Einen Satz aus f lesen und in r ablegen
```

1. Erläutern Sie wie ein deratiges Programm mit einem System wie Windows–NT oder Unix kommuniziert, das nur einen einzigen festen Satztyp kennt. Welchen Anteil haben Compiler, Standardbibliothek und Betriebssystem an der Umsetzung von `R` auf den Satztyp des Systems. Welcher Code wird vom Compiler erzeugt, welche Informationen benötigt er dazu über

   - die speziellen Anwendung (das Programm),
   - das Betriebssystem,
   - die Hardware (welche: Platte, CPU, ..)?

   Welche Systemroutine (z.B. von Unix) wird vom übersetzten Programm wie aufgerufen?
2. Wird eine Datei die von einer Anwendung erzeugt wird, wohl von einer anderen gelesen werden können, wenn:

   - unterschiedliche Compiler auf dem gleichen System,
   - unterschiedliche Hardware (z.B. Intel– oder RISC–CPU) bei gleichem Betriebssystem,
   - unterschiedliche Betriebssysteme bei gleicher Hardware,
   - unterschiedliche Hardware und unterschiedliche Betriebssysteme

verwendet werden?

3. Wie wird das Programmfragment von oben in C++ formuliert?
4. Wie kommuniziert das übersetzte C++–Programm mit dem Betriebssystem, welche Informationen benötigt er dazu über

   - die speziellen Anwendung,
   - das Betriebssystem,
   - die Hardware?

   Welche Systemroutine (z.B. von Unix) wird vom übersetzten Programm wie aufgerufen?

Aufgabe 6

1. Was ist ein Port?
2. Was treibt ein "Treiber" und was kontrolliert ein "Controller"? Handelt es sich um Hardware oder um Software?
3. Muss ein DMA–Controller seine Aktivitäten (welche sind das?) mit der CPU in irgendeiner Form abstimmen, oder operieren beide – abgesehen vom Start und Ende der DMA–Aktivität – völlig unabhängig?
4. Nehmen Sie an, dass bei der Abarbeitung einer Anwendung der Benutzer die linke Maustaste drückt um eine bestimmte Funktion zu aktivieren. Ist das Betriebssystem davon in irgendeiner Art betroffen?
5. Ist die Maus ein blockorientiertes Gerät?