Prof. Dr. Th. Letschert
FB MNI
30. Juni 2002

Übung 5 - Betriebssysteme I

Aufgabe 1

1. In welchem Adressraum liegen die Portadressen?
2. Wo befindet sich der Controller der Maus, der Festplatte, des Druckers, des Monitors?
3. Beim Polling beobachtet die CPU eine Port solange bis erkennbar ist, dass Daten eingetroffen sind und holt diese dann ab. Erläutern Sie was genau unter “Beobachten” und “Abholen” zu verstehen ist.
4. Welche Alternative(n) gibt es zu Polling?
5. Was bedeutet der Begriff “DMA”?
6. Wie löst man das Problem, dass ein Treiber Bestandteil des Betriebssystems ist, aber bei der Auslieferung eines Betriebssystems nicht klar ist, welche Hardwarekomponenten von ihm aktuell zu bedienen sind?
7. Nennen Sie ein Betriebssystem, das im Quellcode ausgeliefert wird und eines das nicht im Quellcode ausgeliefert wird. Welchen Vorteil hat es, wenn der Quellcode vorliegt?
8. Warum werden “die Bits” auf der Platte in Sektoren aufgeteilt?
9. Ein Sektor enthält einen Fehler–korrigierenden Code, also eine Art Prüfsumme, mit der kleinere Speicherfehler korrigiert werden können. Wer (Anwendung, Betriebssystem, Treiber, etc.) korrigiert die Fehler?
10. Warum werden Blöcke und Sektoren unterschieden? Was ist das überhaupt?
11. Welche Zugriffsart haben Dateien in Unix und Windows? Was versteht man unter Zugriffsart?
12. Welche Satzgröße haben Unix und Windows?
13. Der Systemaufruf `write` hat folgende Aufrufschnittstelle (POSIX–System):

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

    Stecken hierin irgendwelche Inforamtionen über den Satztyp?
14. Kann in Unix beim Öffnen oder Erzeugen einer Datei zwischen einer Text– und einer Binärdatei unterschieden werden?
15. Kann in C++ beim Öffnen oder Erzeugen einer Datei zwischen einer Text– und einer Binärdatei unterschieden werden?

Aufgabe 2

Welches Dateikonzept und welche elementaren Dateioperationen hat C++? Erläutern Sie dies an einem Beispiel in dem 50 `Structs` vom Typ

```
struct Dozent {
  char         name[20];
  char         fach[5];
  unsigned int personalNr;
};
```

in einer Datei gespeichert werden und dann der 13. und der 17. `Dozent` ausgelesen werden.

1. Welches Satzkonzept hat bei diesem Beispiel: die Anwendung, die C++–Standardbibliothek, das Betriebssystem.?

2. Geht C++ von einer bestimmten Zugriffsart aus: welche Zugriffsart hat ein `fstream`–Objekt?

3. Wie speichert man Daten im Binär– oder im Text–Format? Worin besteht der Unterschied? Kann man in C++ zwischen Binär und Text–Dateien unterscheiden?

4. Sie wollen Daten (numerische Werte als Int– oder Float–Zahlen) in einem System auf Datei schreiben und diese Datei auf einem anderen System lesen. Auf was müssen Sie achten: die Sprachen in der die lesende und die schreibende Anwendung programmiert wurde, das Betriebssystem unter dem sie laufen, die Byteordnung der Systeme, ...?

5. Worin besteht generell der Unterschied zwischen einer Text– und einer Binärdatei bei den Systemen mit einem Byte als Satztyp? Wie macht sich dieser Unterschied in der praktischen Programmierarbeit bemerkbar?

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

   ```
   TYPE Dozent = RECORD ....;    { Typdefinition: Dozent ist ein Typ }
   TYPE File   = FILE OF Dozent; { Typdefinition: File ist der Typ }
                                   der Dateien mit Satztyp Dozent }
   f : File;                     { Var.-def.: f ist eine Datei mit Satztyp Dozent}
   d : Dozent;                   { Var.-def.: d ist eine Var. vom Typ Dozent}
   ...
   read(f, d);                   { Einen Satz aus f lesen und in d ablegen}
   ```

   Welche Arbeit wird dabei der Anwendung erspart und von wem (Bibliothek, Betriebssystem) wird sie übernommen? Welche Probleme ergeben sich eventuell dabei?

7. Angenommen Sie schreiben eine Anwendung – z.B. einen Text–Editor –, die Dateien mit ASCII–Text beschreibt und liest. Ihre Anwendung soll in C++ implementiert werden und sowohl auf Windows–, als auch auf Unix–Systemen laufen. Die erzeugten Dateien sollen zwischen den Systemen beliebig ausgetauscht werden können.

   Behandeln Sie die Dateien als Text– oder als Binärdateien (in Sinne von C++)?

Aufgabe 3

Bei einer *Memory–Mapped* Datei wird die Datei in den Adressraum des Anwendungsprozesses abgebildet. In Unix–Systemen gibt es beispielsweise den Aufruf

```
mmap(void * Adresse,...int Dateideskripor,...)
```

mit dem der Inhalt der durch den `Dateideskriptor` angegebene Datei ab der `Adresse` im eigenen Adressraum angesprochen werden kann.

Lesen Sie die Manual–Seite zu diesem Aufruf.

Welche Vorteil bietet `mmap` der Anwendung?

Erläutern Sie wie dieses Konzept in einem System mit Paging (relativ) einfach und effizient implementiert werden kann.

Aufgabe 4

1. Was versteht man unter einem Dateisystem?
2. Was ist eine Implementierung eines Dateisystems. Was gehört zur Implementierung eines Dateisystems und wo befindet sich diese Implementierung?
3. Gibt es verschiedene Arten von Dateisystemen und wenn ja, wodurch unterscheiden sich diese Arten?
4. Eine FAT ist eine Tabelle, deren Inhalt eine Abbildung ist, die Blocknummern die Nummer des nächsten Blocks in der gleichen Datei zuordnet. Erläutern Sie dieses Prinzip an einem einfachen Beispiel. Warum ist die Speicherung mit FAT–Tabelle eine Variante der verketteten Dateiorganisation.
5. Erläutern Sie den Sinn der gestaffelten Indirektion bei der Dateiorganisation von Unix–Dateisystemen.
6. 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?
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. 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 überhaupt?