Prüfungsprotokoll Praktische Informatik (Kerngebiet)

Prüfer:	Dr. Martin Lehmann (Tel. 42883-2423 / Fr. 15-16 Uhr)
Kandidat:	cand. Dipl. Inform. Christoph Haas
Datum:	21.10.1999 / 10:00 Uhr
Wetter:	verdammt trübe und kalt
Ort:	Büro von Dr. Lehmann
Note:	1,7

Prüfungsstoff:

- Betriebssysteme (Andrew S. Tanenbaum: „Moderne Betriebssysteme“, 2. Auflage, außer Implementationsdetails und Fallbeispielen)
- Verteilte Systeme (dito)
- Programmiersprachen (Kenneth C. Louden: „Programmiersprachen“, Kapitel 1,4,5,6,7,8)
- Datenbanken (C.J.Date: „Introduction to database systems“, 5. Auflage, Kapitel I-III)
  ➔sehr schwafelig – großzügig lesen!
  ➔anderer Schwerpunkt als 6. und 7. Auflage!

Vorbereitung

Nach einem Telefonat mit Dr. Lehmann sollte ich mir 1200 Seiten Stoff aus gängigen Büchern zusammensuchen – die Inhalte seien ohnehin fast dieselben. Diese Aufstellung der Bücher und Kapitel ist dann zur Prüfung mitzubringen. Nach Lehmanns Aussagen habe er die 1200 Seiten als Maß genommen, damit der Student nicht auf Idee kommt, selbige auswendig zu lernen. Außerdem seien einige Autoren sehr wortinflatorisch. Der Stoff soll nicht mehr als 6 Wochen intensiver Vorbereitung bedürfen. (Laut einem anderen Protokoll hat ein scheinbar geistig verwirrter Student ein ganzes Jahr gebraucht. Friede seiner Seele.)

Wichtig ist (auch das wurde mir im persönlichen Gespräch mitgeteilt), dass man sich darauf vorbereitet, viel selbst zu reden. Im Gegensatz zu anderen Prüfungen geht es nicht um Frage-Antwort-Spielchen. Vielmehr gibt Dr. Lehmann einen Themenkomplex vor (relationale Datenbanken, Speicherverwaltung, zentrale versus verteilte Systeme, Scheduling), über den man dann solange reden können sollte, bis er einen abwürgt. Das Ganze artet zwar nicht in einen 30-minütigen Monolog aus, aber eine kleine Gliederung wichtiger Themen sollte man sich schon vorher überlegt haben. Ich habe die aus den Büchern übernommen, um die Sachverhalte sinnvoll strukturiert wiederzugeben.

Die Terminvergabe ist höchst dynamisch. Sofern er nicht im Urlaub ist, reicht eine Absprache 2 Wochen vorher. Ängstlicherweise habe ich ihm meine Literaturliste noch einmal vorher gezeigt.

Meine Vorbereitungszeit war etwa 3 Monate. Einen Monat habe ich gemütlich die Bücher gelesen (1-2 Tage die Woche). Einen Monat habe ich intensiver die Inhalte der Bücher in eigene schriftliche Worte gefasst (das lehrt automatisch einiges). Und erst die letzten beiden Wochen habe ich aufgrund meiner Zusammenschrift angefangen, abgegrenzte Algorithmen, Argumente, Ausdrücke und Abstraktionen auswendig zu lernen (4-6 Stunden/Tag über selbigen verteilt). In dieser Zeit habe ich mir Urlaub vom Job genommen - es war auch so stressig genug.

Ich habe leider bei keiner Prüfung zuhören können und habe nur aufgrund kommilitonischer Empfehlung Dr. Lehmann als Prüfer gewählt – selbiges aber nicht bereut. Meine Umgebung hat mich mehrfach stundenlang abgefragt. Schließlich wirkte die Prüfung fast nur noch wie eine weitere Abfragerunde. ;-)

Unterlagen

Meine Lernunterlagen (ca. 60 Seiten) gibt es in ästhetisch ansprechender digitaler Form unter ftp://ftp.newton-lifestyle.de/pub/uni als Postscript zum downloaden. Die Dokumente sind Protokollware (wenn Du sie benutzt, musst Du auch ein Protokoll schreiben) und unterliegen ansonsten unter der GPL (Gold Pressed Latinum). Wichtig ist aber, dass Du selbst auch eigene Formulierungen findest. Wer einen Sachverhalt selbst darstellen kann, der hat ihn auch verstanden. Und in der Prüfung zählt nur, was man darstellen kann – nicht, was man verstanden hat. Und wenn Du einen Tag vor der Prüfung das Gefühl hast, alles vergessen zu haben, dann wird alles gut. ;-)

Gesamteindruck

Das Vorleiern formalster Definitionen ist hier dem Beweis des Verständnisses gewichen. Aus der eigenen Praxis kann man hier notfalls auch mal aus dem Nähkästchen plaudern und das geplagte Hirn freut sich über jeden Bezug zur eigenen informatischen Praxis. Nicht immer habe ich meinen roten Faden konsequent verfolgt. Aber auch einige Minuten Frage-Antwort-Spielchen haben die Stimmung nicht verdorben. Ich konnte mich gut konzentrieren und reden und denken gleichzeitig.

Prüfungsablauf

Ein grauer Tag in meiner Geschichte, allerdings nur rein wetterlich. Ich liebe Prüfungsvorbereitungen zu ungemütlichen Jahreszeiten, man kann wenigstens während der Vorbereitungszeit draußen nicht viel anderes machen. Mein Mitlerner war als Zuhörer

auch live mit dabei. Meine Unterlagen eine Stunde vor der Prüfung noch einmal durchzulesen war reine Zeitverschwendung. Ich muss mir das abgewöhnen.

Dann suchen Sie sich aus Ihrer Übersicht mal ein Thema aus und fangen Sie damit mal an.

[Das ist ja leichter als erwartet. Für diesen Fall hatte ich eine gute Gliederung meines Lieblingsthemas ‚Speicherverwaltung' im Kopf.] Dann nehme ich mal Betriebssysteme und dabei die Speicherverwaltung. Früher gab es lediglich den Einprogrammbetrieb. Es gab einen Bereich für das Betriebssystem und der Rest konnte vom einzigen Prozess benutzt werden. Dann hat man sich überlegt, dass es sinnvoll ist, mehrere Prozesse gleichzeitig zuzulassen, weil viele Prozesse einen hohen I/O-Anteil haben und die CPU dann untätig wäre. [Ich wollte noch die nette Formel bringen, aber ich hatte Angst, meine Faden zu verlieren.] Im nächsten Schritt ging man dann zu fixierten Partitionen über. Das führte zu einer hohen internen Fragmentierung, erlaubte aber grundlegendes Multitasking. Schließlich nahm man variable Partitionen, bei denen Prozesse exakt Speicher allokieren können, was zu keiner internen, aber einer gewissen externen Fragmentierung führte. Und dann gibt es noch einige Algorithmen, mit deren Hilfe das Betriebssystem den Prozessen Speicherbereiche zuweist, das sind die Speicherzuteilungsalgorithmen. [Leider führte ich hier auch schon die Seitenrahmen ein, was im allgemein Modell nicht richtig war. Seitenrahmen gibt es nur bei virtuellem Speicher. Ansonsten gab es soweit keine Zwischenfragen sondern eher bei jedem Satz ein zustimmendes Nicken.]

Sie haben hier in der Zeichnung das Betriebssystem in einen eigenen Bereich gezeichnet. Wie ist denn das mit Puffern, liegen die auch in diesem statischen Bereich?

[Hier konnte ich nur schwimmen und vermutete, dass das Betriebssystem im Bereich der Partitionen noch Speicher für Puffer belegt. Ich konnte nicht klar ersehen, ob das nun richtig oder falsch war.]

Dann kommen Sie doch mal zum Paging.

Der gesamte Speicher (Hauptspeicher + Auslagerungsspeicher) wird als virtueller Speicher adressiert. Dabei wird der virtuelle Speicher in Seitenrahmen unterteilt, in denen jeweils Speicherseiten liegen können. Die MMU (memory management unit) sitzt zwischen CPU und dem Speicherbaustein auf dem Speicherbus und übersetzt virtuelle Adressen in reale Speicheradressen, sofern sich die Seite im Hauptspeicher befindet. Ist die Seite auf der Festplatte, dann wird anhand von externen Softwaretabellen diese Seite durch eine Exception vom Betriebssystem eingelagert und – falls kein Platz ist – eine alte Seite wieder ausgelagert. [Er wollte nichts näheres zu dieser Exception wissen. Gut!] Die Verwaltung dieses Speichers übernimmt die Seitentabelle. Dort steht die Adresse, wo die Seite zu finden ist und eine Reihe von Flags. Es gibt einstufige und mehrstufige. Bei mehreren Stufen ist die Tabelle kleiner, wenn nur weit entfernte Segmente im Speicher belegt sind. [Beispiel mit 32 Bit aufgemalt und die Ersetzung der Rahmennummer und des Offsets erklärt.]

Was ist denn, wenn Sie mehr als 32 Bit für die virtuelle Adresse nehmen?

Dann nimmt man mehr Bits für die Seitentabellen.

Ich meinte etwas anderes. Nehmen Sie mal an, sie haben keine 32 Bit sondern 64 Bit. Dann können Sie ja mehr als 2 MB damit adressieren.

[Geht es jetzt um invertierte Seitentabellen? Und seit wann adressieren 32 Bit 2 MB?] Entschuldigung, aber 32 Bit können 2 GB adressieren, nicht 2 MB.

Wirklich? [Zum Beisitzer…] Was meinen Sie dazu?

[Der Beisitzer pflichtete mir bei. Sehr schön.]

Nun gut, auf jeden Fall ist das eine ganze Menge. Was ist, wenn Sie jetzt viel mehr Hauptspeicher haben als der virtuelle Speicher groß ist?

Normalerweise ist der virtuelle Speicher nicht kleiner als der Hauptspeicher. Ist der Hauptspeicher groß genug, dann kann man natürlich auf virtuellen Speicher verzichten und das ganze Paging und Swapping. Das ist eher ein Artefakt aus Zeiten des Speichermangels.

Okay, weiter…

Wie gesagt, es gibt die Seitenersetzungsalgorithmen, die im Hauptspeicher Platz zum einlagern schaffen sollen. Da gibt es NRU (Not recently used), das sich um das Referenced-Flag in der Seitentabelle kümmert. Dann gibt es LRU (Least recently used). LRU unterteilt Speicherseiten in vier Kategorien, wobei… [Schade, unterbrochen worden. Ich hätte gerne noch was dazu gesagt, wie man zu Klasse-1-Seiten kommt.]

Das reicht soweit. Welche Algorithmen gibt es noch?

Es gibt noch FCFS. Äh, ich meine natürlich FIFO. FCFS steht ja für First-Come-First-Served und das passt hier natürlich nicht.

Naja, ist ja praktisch dasselbe.

Und dann gibt es noch den Second-Chance-Algorithmus, der eine Liste von Seiten verwaltet und nach jedem Durchlauf das Referenced-Flag löscht.

Und es gibt noch den Uhr-Algorithmus, der ähnlich wie Second-Chance funktioniert.

Der Unterschied ist ja auch nicht so ganz klar.

[Gut, war ich also nicht zu blöd dazu.]

Okay, das reicht soweit. Dann haben Sie hier auf Ihrem Zettel Programmiersprachen stehen. Hier ist ein Kapitel Datentypen. Sagen Sie doch mal was dazu.

Datentypen sind Mengen von Werten mit darauf definierten Operationen. Man kann aus den zur Verfügung stehenden Basistypen neue Typen erzeugen, weil sie ja Mengen sind – auch hier kann man Mengenoperationen verwenden. Basistypen sind so was wie Integer, Real oder Character.

Gibt es denn mehrere Versionen von Integer?

[Äh… Endian? Nein. Verschiedene Programmiersprachen? Nein. Äh…] Öhm… also…

Es gibt doch mindestens zwei Arten von Integer.

Ach so, ja, unsigned-integer und signed-integer.

Genau, und was ist da mit Typkompatibilität?

Kompatibel bedeutet, dass man die beiden Typen miteinander verknüpfen kann, das ist bei Integer natürlich möglich. [Der Gedanke, dass das nicht stimmen konnte, kam mir zu spät.]

Wirklich? Was ist denn, wenn Sie signed-integer in unsigned-integer zuweisen? Was machen Sie denn bei negativen Werten?

Ach so, das kommt auf den Übersetzer an. Entweder schlägt zur Compile-Zeit schon die Typüberprüfung zu und meldet einen Fehler. Oder man – verschiebt das Programm bis zur Laufzeit und hofft, dass keine negativen Werte zugewiesen werden.

Ja. Was sind denn dabei Ausnahmen?

Ausnahmen (exceptions) werden ausgelöst, wenn zur Laufzeit solche Fehler wie eben auftreten. Es gibt dann eine Ausnahmebehandlung (exception handler), die sich des Problems annimmt.

Und die kann der Programmierer vorgeben?

Entweder kommt der von der ausführenden Umgebung oder der Programmierer kann einige Laufzeitfehler selbst abfangen und eigene Handler einbinden.

Noch mal zu den Datentypen. Was gibt es denn da für Operationen?

Alles mögliche. Schnittmenge, Teilmenge, kartesisches Produkt. Man kann z.B. eine Vereinigung durch eine Union in C ausdrücken. [Beispiel aufgeschrieben.]

Und was ist mit so was wie Records?

Das sind dann kartesische Produkte. [Struct in C aufgeschrieben.]

In C kennen Sie sich ja ganz gut aus. Was ist denn hier mit Funktionstypen?

[Argl… das gab's doch nur in Modula, oder?] Ähm, ja, also, bestimmt, denke ich. Man könnte ja, äh, nein.

Was ist denn ein Zeiger?

[Hey, Zeiger auf Funktionen. Brilliant!] Zeiger zeigen auf Datentypen bzw. Speicheradressen.

Kann man denn Zeiger auf Funktionen zeigen lassen?

Ja, schon.

Könnten Sie hier nicht Prototypen benutzen? Was ist denn ein Prototyp?

[Erst hören, dann denken, dann antworten. Aufgrund der Missachtung dieser Grundregel analysiert mein Hirn, was zum Henker Inlines mit Zeigern zu tun haben sollen. Inlines sind natürlich etwas völlig anderes als Prototypen.] Man kann Programmteile definieren, die überall im Kontext benutzt werden können.

Nein, das ist es nicht ganz.

[Ich würde mal sagen, das ist es nicht im Gerinsten.]

Auf was würden Sie denn Zeiger zeigen lassen? Auf die Startadresse? Oder auch auf die Parameter?

[Zeiger auf Parameter? Käse.] Nein, die Zeiger zeigen auf den Anfang der Prozedur bzw. Funktion. Damit der Übersetzer aber weiss, welche Parameter eine Funktion übernehmen kann, muss man sie einmal im Deklarationsteil als Prototyp definieren. [Endlich wusste ich wieder, was Prototypen sind.]

Ja, ganz genau. Und Array, was sind das?

Ich nehme mal was aus Modula-2. [Gefiel ihm, dass ich nicht nur C konnte. Die Freude währte allerdings nicht lange.]

TYPEDEF x = ARRAY [1…

Typedef in Modula? Naja, ich weiß schon, was Sie meinen.

Naja, und hier kann man den einzelnen Stellen des Arrays Werte eines bestimmten Datentyps zuweisen. [Wie lange ist es her, dass ich Modula-2 programmiert habe? Merke: Typedef war definitiv falsch.]

[Zum Beisitzer] Wieviel Zeit haben wir noch?

Beisitzer: etwa eine Minute

[Tatsächlich hatte er schon zwei Minuten überzogen, aber ich hatte keine Uhr und freute mich riesig, dass gleich Schluss sein würde.]

Na, dann suchen Sie sich doch noch mal ein kleines Thema aus Ihrer Liste hier aus. Aus einem Bereich, den wir noch nicht hatten. Also verteilte Systeme oder Datenbanken.

[Datenbanken? Nein, danke.] Dann nehme ich mal Gruppenkommunikation.

Gut, warum benutzt man denn Gruppenkommunikation?

Man kann damit mehrere Systeme als eine abstrakte Einheit auffassen und ihnen Nachrichten zukommen lassen. Es gibt offene und geschlossene Gruppen.

Und warum macht man das?

Häufig gibt es funktional zusammenhängende Systeme, die nur untereinander kommunizieren. Dann kann man mit Unicast, Broadcast und Multicast arbeiten. Das kann man zum Beispiel für Prozessorzuteilungsalgorithmen in einem Pool benutzen. [Bitte nicht weiterfragen, das Stichwort ‚Prozessorzuteilung' war dumm gewählt.]

Und warum nimmt man nicht immer Unicast?

Wenn man einen Netzwerkbus hat, kann man mit einem einzigen Datenpaket alle Systeme erreichen. Bei Unicast muss man das Paket an jedes System einzeln schicken.

Ist das denn ein Problem? Was passiert denn bei Ihrer Prozessorzuteilung, wenn ein Rechner nicht antwortet?

Das wird ignoriert, man bekommt ja von den anderen Systemen eine Antwort. [Oh, Gott, war das geraten. Und dann auch noch grob falsch.]

Was? Nein, so geht das ja nun nicht. Was macht man denn, wenn ein Rechner nicht antwortet?

Wollen Sie auf den atomaren Broadcast hinaus?

Ja, sowas in der Richtung.

Ach so, ja man kann atomaren Broadcast einsetzen, damit sich die einzelnen Prozesse nicht darum kümmern müssen, ob eine Nachricht ankommt oder nicht. Das wird garantiert.

Gut, hören wir am besten auf.

[Merke: der gesamte obige Dialog ist natürlich nicht vollständig. Meine Gedankengänge sollen Dich nur belustigen und die ganze Situation und die Atmosphäre beschreiben.]

Das Tribunal

Nachdem ich mich noch vor der Tür freute, dass ich höchstwahrscheinlich nicht durchgefallen bin (ich lerne immer auf ‚durchkommen' – nie auf eine gute Note), wurde mir vom hohen Gericht eröffnet: „Sie haben uns die Notenfindung nicht gerade einfach gemacht. Aber wir konnten uns auf eine 1,7 einigen. Eine 1,3er-Prüfung ist es nicht ganz gewesen." Na, bitte! 1,7 fand ich doch eine tolle Leistung.

Post Scriptum

Ich habe einige Teile dieses Protokolls schon vor meiner Prüfung fertiggestellt. Es hat mir kein Unglück gebracht und mich letztendlich motiviert, das Protokoll auch abzugeben – das hat noch einmal zwei Wochen gedauert, bis ich mich endlich aufgerafft habe. Die große Erleichterung, die Prüfung endlich hinter sich zu haben, kann einen sonst von dieser Pflicht ablenken. Danke auch an Claas für die regelmäßige Vorbereitung. Möge Deine Prüfung auch erfolgreich verlaufen.