Übungsaufgaben zu Grundlagen: Betriebssysteme und Systemsoftware

Katharina Bogad

24. Februar 2018

Wichtiger Hinweis: Diese Aufgaben wurden von mir (Katharina) in meiner Freizeit erstellt. Sie wurden nicht von der Übungsleitung quergelesen und haben mit dem offiziellen Modul nichts zu tun. Ich habe hier versucht eine Reihe an zusätzlichen Übungsaufgaben bereit zu stellen, die den Stoff aus Grundlagen: Betriebssysteme und Systemsoftware abdecken. Natürlich bin auch ich weder perfekt noch allwissend, es kann daher durchaus sein, dass sich ein oder zwei Fehlerchen eingeschlichen haben. Falls ihr einen findet, erbitte ich um Meldung an `bogad@in.tum.de`. Danke!

Die Aufgaben sind von unterschiedlichem Schwierigkeitsgrad. Eine kleine Hilfe zur Einschätzung sind Chilishoten - je mehr Chilis eine Aufgabe hat, desto würziger (also schwerer) ist sie. Generell gilt, dass Aufgaben mit vier oder mehr Chilishoten wahrscheinlich zu schwer für eine Klausur sind. Sie sind dennoch hier drin, da es Knobelaufgaben für Leute, die ein tiefes Verständnis vom Stoff aufbauen wollen sind. Soll's geben ;)

Diese Aufgaben sind – zusammen mit den Lösungen – auf meiner Homepage https://hacked.xyz/ verfügbar. Bitte stellt sie nicht auf unistuff oder vergleichbare Plattformen, da diese Sammlung ständig erweitert wird und niemandem geholfen ist, wenn alte Versionen herumflattern. Diese Aufgabensammlung ist auf keinen Fall umfassend und ersetzt nicht den Besuch an Vorlesung oder Übung. Insbesondere enthalten sie derzeit kaum "Lernfragen". Bitte schaut euch die Lösungen zu den Übungsblättern nochmal genau an! Wir haben in den Übungen einiges an Theorie durchbesprochen. Unterschätzt das nicht.

Viel Erfolg für die Klausur!

1 Betriebsmodi

𝒻 Entscheiden Sie jeweils, ob der angegebene Prozess im Kernelspace oder im Userspace stattfindet.

Prozess / Aktion	Userpace	Kernelspace
Grafikkartentreiber	☐	☐
Zugriff auf beliebigen Speicher	☐	☐
Berechnen von π	☐	☐
Alle Interrupts deaktivieren	☐	☐
MD5-Hashe knacken	☐	☐
Speicher allokieren	☐	☐

2 Prozesse und Threads

𝒻 𝒻 In der Vorlesung wurden Threads als leichtgewichtige Prozesse eingeführt. Sie sind Teil eines übergeordneten Prozesses. Was bedeutet dies für den Adressraum des Prozesses / der Threads? Sind Szenarien denkbar, in denen diese Implikation zum Problem werden kann?

3 POSIX-Funktionen

𝒻 𝒻 𝒻 Betrachten Sie folgenden Auszug aus dem `Linux Programmers Manual`. Beschreiben Sie anschließend kurz die Funktion memmove und ihre Parameter. Welchen Unterschied gibt es zu der Ihnen wohlbekannten Funktion `memcpy(3)`?

```
MEMMOVE(3)       Linux Programmer's Manual       MEMMOVE(3)

NAME
       memmove - copy memory area

SYNOPSIS
       #include <string.h>

       void *memmove(void *dest, const void *src, size_t n);

DESCRIPTION
       The  memmove()  function  copies n bytes from memory
       area src to memory area dest.  The memory areas  may
       overlap:  copying takes place as though the bytes in
       src are first copied into  a  temporary  array  that
       does not overlap src or dest, and the bytes are then
       copied from the temporary array to dest.

RETURN VALUE
       The memmove() function returns a pointer to dest.

ATTRIBUTES
       For an explanation of the terms used  in  this  sec-
       tion, see attributes(7).
```

```
       [...]
CONFORMING TO
       POSIX.1-2001, POSIX.1-2008, C89, C99, SVr4, 4.3BSD.

SEE ALSO
       bcopy(3),    memccpy(3),    memcpy(3),    strcpy(3),
       strncpy(3), wmemmove(3)

COLOPHON
       This page is part of release 4.09 of the Linux  man-
       pages project.  A description of the project, infor-
       mation about reporting bugs, and the latest  version
       of     this     page,     can     be     found    at
       https://www.kernel.org/doc/man-pages/.

GNU                        2015-08-08              MEMMOVE(3)
```

4 C-Programmierung

𝒻𝒻 Finden und beschreiben Sie drei Fehler in folgendem Codeschnipsel:

```
1  #include<stdio.h>
2
3  int main(int argc, char** argc) {
4          char buf[50];
5
6          read(fileno(stdin), &buf, sizeof(buf));
7          printf("Hallo " + buf);
8          putc(0x0A);
9
10         gbsisttoll();
11
12         return 0;
13 }
14
15 void gbsisttoll() {
16         puts("GBS ist toll!\n");
17 }
```

5 Petrinetze

(a) 𝒻𝒻 Der Freisinger Stadtbusbahnhof besteht aus fünf Haltestellen. Jede Haltestelle bietet Platz für zwei Busse in Standardlänge oder einen Gelenkbus. Zur Vereinfachung können Sie annehmen, dass der morgendliche Berufsverkehr nicht existiert und somit nur die erste Haltestelle von einem Gelenkbus zum Flughafen München bedient wird; an allen anderen Haltestellen halten Busse in Standardlänge. Betrachten Sie folgendes Bild der Situation:

Außerdem gibt es noch weitere Einschränkungen:

- Da die Ein/Ausfahrt in einer Kurve liegt, kann sie immer nur von einem Bus gleichzeitig belegt werden.
- Es kann immer nur jeweils ein Bus im rechten Teil, in der Wendekurve und im linken Teil gleichzeitig sein.
- Im Gegensatz zur Realität machen unsere Busse nie Pause. Parkplätze sind daher nicht notwendig, jeder einfahrende Bus fährt immer in eine Station.

Modellieren Sie das Petrinetz, dass die oben genannte Situation beschreibt. Beantworten Sie auch - ggf. unter Zuhilfenahme eines Erreichbarkeitsgraphen - folgende Fragen:

1. Kann das Netz verklemmen? Wenn ja, wo?
2. Können einzelne Transitionen verhungern?

(b) 𝒟 𝒟 𝒟 𝒟 Im Folgenden soll nun basierend auf dem Problem aus (a) ein Producer-Consumer-Modell erstellt werden. Nehmen Sie an, dass ankommende Regionalzüge und S-Bahnen Fahrgäste Produzieren (proucer), die von den Bussen nach Hause gefahren werden (consumer). Achten Sie insbesondere auf korrekte Kapazitäten!

6 Prozesse, Dämonen und Zombies

𝒻 𝒻

(a) Erläutern Sie kurz die Bestandteile eines Prozesses.

(b) Wann wird ein Prozess ein Dämon? Welche Schritte sind notwendig?

(c) Wann wird ein Prozess ein Zombie?

7 Semaphoren

𝒻 Da die Fragestunde ziemlich lange dauert, bestellen einige Studenten bei der Pizzeria Pappeloni Pizza. Der Pizzabäcker Paolo produziert immer eine Familienpizza und eine normale Pizza gleichzeitig. Im Pizzaofen ist Platz für insgesamt 6 Pizzen, wobei eine Familienpizza soviel Platz wie zwei normale Pizzen benötigt. Seine Söhne Enrico und Pietro verpacken die Pizzen dann in Kartons. Durch die hohe Nachfrage zur Prüfungszeit gibt es allerdings schon erste Materialschwächen: Es ist nur noch ein Pizzaschieber einsatzbereit, den sich Paolo, Enrico und Pietro teilen müssen.

Modellieren Sie das angegebene Szenario mit Semaphoren. Beachten Sie dabei folgende Hinweise:

- Enrico und Pietro können den Pizzaschieber nur allokieren, wenn auch eine Pizza im Ofen ist, die sie abholen können.
- Enrico verpackt nur Familienpizzen, Pietro nur normale Pizzen.
- Paolo kann den Pizzaschieber nur allokieren, wenn er alle Pizzen im Ofen unterbringen kann.
- Zum Zeitpunt t_0 ist der Ofen leer, aber auf 230°C vorgeheizt.
- Initialisieren Sie geeignete Semaphoren selbstständig.
- Zur Vereinfachung gehen wir von einem Wunderofen aus, der Pizzen sofort nach dem Ablegen saftig heiß und mit Knusperboden fertig werden lässt.

Guten Appetit!

```
Prozess Paolo          Prozess Enrico         Prozess Pietro
{                      {                      {
  while (TRUE)           while (TRUE)           while (TRUE)
  {                      {                      {
```

```
  }                    }                    }
}                    }                    }
```

8 Banker's Algorithmus

Hinweis: Diese Aufgabe kam dieses Jahr nicht in der Übung vor.

🌶 [1] Ist der angegebene Systemzustand stabil? Führen Sie den Banker's Algorithmus aus, um - falls möglich - eine Ausführungsreihenfolge der Prozesse zu bestimmen.

$$n = m = 4, c = \begin{pmatrix} 1 & 2 & 5 & 0 \\ 0 & 1 & 7 & 3 \\ 4 & 2 & 8 & 0 \\ 1 & 3 & 3 & 7 \end{pmatrix}, r = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 2 & 4 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 5 & 2 \end{pmatrix}, e = \begin{pmatrix} 6 & 10 & 23 & 11 \end{pmatrix},$$
$$a = \begin{pmatrix} 0 & 2 & 0 & 1 \end{pmatrix}$$

9 Scheduling-Strategien

Für die folgende Aufgabe seien drei Prozesse p_1, p_2, p_3, zwei Kernel-Level-Threads von p_1, k_1 und k_2 sowie zwei User-Level-Threads von p_3, u_1 und u_2 gegeben. Der Prozessvektor ist damit $\vec{p} = (k_1, k_2, p_2, p_3, u_1, u_2)$. Als Ankunftszeiten seien $\vec{a} = (0, 3, 2, 4, 8, 4)$, als Rechenzeiten $\vec{r} = (3, 4, 2, 3, 2, 2)$ gegeben. Nehmen Sie an, dass ein Schedulingvorgang (egal ob Kernel- oder Userland) eine Zeiteinheit benötigt; die zum Dispatchen benötigte Zeit ist vernachlässigbar. Das Userland-Scheduling zählt **nicht** zur Runtime des Prozesses. Als Tie-Breaker setzen Sie die Prozess-ID an, wobei niedrigere Prozess-IDs höhere Priorität haben. Im Userland-Sheduler haben Threads priorität ggü. eigenen Berechnungen. Skizzieren Sie den Ablauf der Prozesse in einem Gantt-Diagram für folgende Sheduling-Arten:

(a) 🌶🌶 **preemptive Round-Robin** mit konstanten Zeitquanti von $t_i = 2$ Zeiteinheiten.

(b) 🌶🌶🌶🌶 **non-preemptive Shortest Job First**. Welches Problem dieser Strategie wird sichtbar?

10 Strategien zur Speicherallokation

🌶 Gegeben ist eine Liste freier Speicherbereiche: 100kB - 250kB - 300kB - 50kB - 400kB. Wie sieht diese Liste nach den Anfragen für 20kB - 300kB - 60kB - 300kB - 40kB aus?

(a) Verwenden Sie die Strategie **First Fit**.

(b) Verwenden Sie die Strategie **Worst Fit**.

[1] Selbst eine Chilischote ist hier eigentlich zuviel...

11 Dateisysteme, I-Nodes

Der Hacker namens 4Chan[2] lädt sich von seiner Lieblingstorrentseite den Film HotFuzz herunter. Da er wie alle Hacker natürlich Linux auf seinem Rechner verwendet, muss der Linux-Kernel nun mit einer Datei der Größe von rund 5GB umgehen.

(a) 🕊🕊 Wie wird diese Datei im Speicher organisiert? Wie viele Blöcke (bei einer Blockgröße von 4KB) werden benötigt?

(b) 🕊 Nehmen wir an, der Film liegt unter `/home/1337h4x0r/Videos/hotfuzz.bdrip.4k.mkv`. Unser Hacker hat im Keller ein NAS stehen, auf das er bequem vom Fernseher aus zugreifen kann. Das NAS sei unter `/mnt/nas` gemountet. Da der Film ziemlich groß ist, und der Hacker ziemlich ungeduldig, legt er der Einfachheit halber eine symbolische Verknüpfung an. Der Bundestrojaner offenbarte uns folgendes directory listing (`ls -la`):

```
...
lrwxrwxrwx 1 1337h4x0r users       7 23. Feb 00:44 hotfuzz.mkv
                  -> /home/1337h4x0r/Videos/hotfuzz.bdrip.4k.mkv
...
```

Warum funktioniert das nicht?

12 Seitentabellen, Adressaufbau

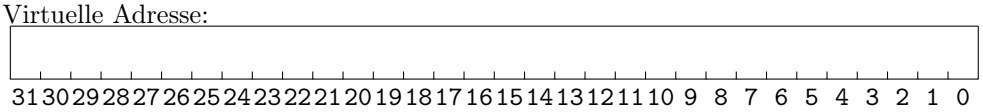
Für die folgenden Teilaufgaben gelten folgende Annahmen:

Die 256 KiB eines byte-Adressierbaren virtuellen Adressraums werden mit einem zweistufigen Pagingverfahren auf Kacheln der Größe $4096 = 0x1000$ abgebildet. Insgesamt stehen dem System 64 KiB physisch adressierbarem Hauptspeicher zur Verfügung.

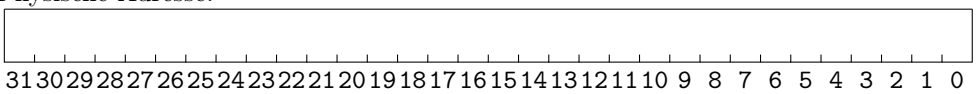
Gehen Sie davon aus, dass nicht auf das Offset innerhalb einer Seite entfallende Bits einer virtuellen Adresse so auf die beiden Paging-Stufen aufgeteilt wird, dass die zweite Stufe doppelt so viele Bits bekommt wie die erste. Wie üblich darf Seitengröße = Kachelgröße angenommen werden. Vereinfachend nehmen wir außerdem an, dass außer den Adressinformationen keinerlei weitere Informationen (present, dirty, ...) kodiert werden.

(a) 🕊 Vervollständigen Sie folgende Grafik, in dem Sie die Aufteilung der Adresse in Offset (innerhalb einer Seite/Kachel) sowie ggf. die Bits der Paging-Stufen markieren. Kennzeichnen Sie die Bereiche mit ihrer jeweiligen Funktion. Ein Kästchen eine Adresse entspricht dabei einem Bit. Streichen Sie Bits, die nicht zur Adresse gehören **deutlich**.

Virtuelle Adresse:



Physische Adresse:



(b) 🕊🕊 Bestimmen Sie folgende Werte:

- Zahl der Seiten im virtuellen Speicher bei Vollbelegung

[2] Symbolbild: http://imgur.com/gallery/DtgL7gb

- Zahl der Kacheln im physischen Speicher bei Vollbelegung
- Einträge in der Tabelle, die für den ersten Schritt der Adressübersetzung verwendet wird
- Größe einer Tabelle, die für den zweiten Schritt der Adressübersetzung verwendet wird

Glückwunsch!

Du hast es durchgeschafft. Hier etwas zur Erfrischung:

Viel Erfolg für die Klausur!