Übungsblatt 5

Abgabe bis spätestens 5.12.2016 in Stud.IP

Aufgabe 1 (5 Punkte)

Unter `/home/ti2/ueb/05` findet ihr ein C-Programm `factorial.c`, das iterativ die Fakultät einer gegebenen Zahl errechnet. Die ausführbare Datei `factorial` wurde mit gcc 4.7.2 auf dem Rechner x21 übersetzt.

a) Ermittelt (z. B. mit gdb) den Bereich im virtuellen Adressraum des Prozesses, den die Funktionen `factorial` und `main` einnehmen.

b) Auf welche virtuellen Adressen im Stacksegment wird in der Funktion `factorial` zugegriffen, wenn der User-Stackpointer `%rsp` bei Eintritt in die Funktion den Wert `0x7fffffffdf60` hat? (Geht für davon aus, dass jeder Zugriff ein vollständiges 32-Bit-Datenwort überträgt.)

c) `printf` erhält als erstes Argument die Adresse einer Zeichenkette im Datensegment. Auf welche virtuellen Adressen wird von `printf` beim Lesen dieser Zeichenkette zugegriffen?

d) Die folgende Tabelle zeigt einen Ausschnitt aus der Page-Tabelle eines laufenden Prozesses, der dieses Programm ausführt. (Nicht aufgeführte Pages befinden sich derzeit nicht im Hauptspeicher.) Berechnet für die unter a) bis c) ermittelten Adressbereiche die physischen Adressen, auf die zugegriffen wird. Legt dazu die Annahme zugrunde, dass sich die Page-Tabelle während dieser Zeit nicht ändert und dass Page-Frame 0 ab Offset 0x1000000 im Hauptspeicher beginnt. Die Page-Größe beträgt 4 KiB.

Page-Nummer	Page-Frame
0	10342
1023	394
1024	17363
1025	16
4195708	99
4195813	100
4294967292	239123
4294967293	2001
4294967294	1394

Aufgabe 2 (2 Punkte)

Wir haben in der Vorlesung das Unix-V7-Dateisystem als Beispiel betrachtet und dort die Inode-Struktur kennengelernt, die vielen späteren Dateisystemen zugrunde liegt. Für diese Aufgabe sollen folgende Randbedingungen gelten:

- Ein Datenblock ist 512 B groß.

- Ein Inode ist 128 B groß.
- Ein Inode enthält direkte Einträge für die ersten zehn Datenblöcke einer Datei sowie jeweils einen Indirektblock, einen Zweifach-Indirektblock und einen Dreifach-Indirektblock.
- Blocknummern und Inode-Nummern werden immer in vier Bytes repräsentiert.

Wieviele Blöcke werden mindestens benötigt, um eine Datei der Größe 33696325 B auf der Platte zu speichern? (Bedenkt, dass Inodes und Indirektblöcke ebenfalls Plattenplatz beanspruchen.)

Aufgabe 3 (3 Punkte)

Ein Programm setzt die folgenden Systemaufrufe ab. Geht davon aus, dass keiner von ihnen fehlschlägt und die Datei `nikolaus.avi` 12 MiB groß ist. Die Datei `meta` existiert, aber ist leer.

```
// aus unistd.h:
// # define SEEK_SET    0   /* Seek from beginning of file.  */
// # define SEEK_CUR    1   /* Seek from current position.  */
// # define SEEK_END    2   /* Seek from end of file.  */

  int f = open("/home/ti2/archive/nikolaus.avi", O_RDONLY);
  int g = open("/home/ti2/meta", O_RDWR);

  /* ... */
  lseek(f, -10000, SEEK_END);
  count = read(f, buf, 4096);
  write(g, buf, count);
```

Es wird mit einem Unix-V7-ähnlichen Dateisystem gearbeitet und es gilt zudem:

- Die Blockgröße der Festplatte beträgt wie die logische Blockgröße 512 Bytes.
- Die Inodes sind 128 Bytes groß. Der erste Inode-Block ist Block 2 der Festplatte.
- Nur der Inode für `/` befindet sich bereits in der Inode-Tabelle, der zugehörige Block 2 befindet sich im Buffer-Cache.
- Die verwendeten Dateien haben die in der folgenden Tabelle bezeichneten Inode-Nummern:

Name	/	home	ti2	archive	nikolaus.avi	meta
Inode	0	36	99	206	12783	112

- Die Inodes sind in den in der folgenden Tabelle angegebenen Blöcken im Dateisystem untergebracht:

Inode	0	36	99	206	12783	112
Block	2	9	2000	3101	50	8521

- Die Verzeichnis-Dateien sind jeweils nur einen Datenblock lang.
- Der Buffer-Cache ist so groß, dass im Rahmen dieser Aufgabe nicht verdrängt werden muss.
- Die Datei `count`

Verfolgt die Abarbeitung der Systemaufrufe und beschreibt, welche Inodes und Datenblöcke von der Festplatte in den Hauptspeicher geladen werden müssen. Ziel ist es zu erarbeiten, wie Ihr

Euch „von oben nach unten“ über die Inodes und Datenblöcke der Verzeichnisse durch dieses Dateisystem bewegen könnt. Anhand der von uns gemachten Angaben könnt Ihr teilweise keine konkreten Plattenblöcke angeben. Es reicht dann aus, wenn Ihr z. B. sagt, „dass der Datenblock der Verzeichnisdatei `tmp` gelesen wird“.

Weitere Aufgaben

1. Warum ist ein perfekter Algorithmus zur Verdrängung von Pages aus dem Hauptspeicher nicht realisierbar? Wie arbeiten die folgenden Algorithmen in etwa:

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

2. In welche dieser Kategorien kann man NRU (Not-Recently-Used) einordnen? Wie arbeitet der Clock-Hand-Algorithmus?
3. 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?
4. Was ist Swapping? Warum wenden auch Paging-Systeme dieses Verfahren an bzw. unter welcher Bedingung?
5. Wie kann man die Vorteile von Paging und Segmentierung kombinieren?
6. Wozu bzw. wo wird bei der Speicherverwaltung häufig ein Assoziativspeicher eingesetzt?
7. Beschreibe kurz die Zugriffsoperationen `open()`, `close()`, `lseek()`, `read()` und `write()` auf ein Unix-Filesystem. Welche Rolle spielt dabei der *Filedeskriptor*?
8. Wie sieht die Struktur des Unix-V7-Dateisystems auf der Platte in etwa aus? Warum erfolgt die Verwaltung der Freispeicherliste über *Indirekt*-Blöcke?
9. Welche Angaben enthält ein Inode? Welche Angaben enthält eine Verzeichnis-Datei (Euch besser bekannt als „Directory“)?

Diese Aufgaben müssen nicht abgegeben werden. Sie dienen als Vorbereitung auf das Fachgespräch und werden im Tutorium besprochen.

Abgabe

Bis 24:00 Uhr am 5.12.2016 digital in Stud.IP und – wenn nicht anders mit Eurem Tutor vereinbart – ausgedruckt in unser Postfach in der MZH-Ebene 6 oder im nächsten Tutorium. Es gelten die vereinbarten Scheinbedingungen (siehe Stud.IP). Bitte beachtet unsere ergänzenden Hinweise ebenda.

Packt die abgaberelevanten Dateien in eine Archivdatei. Die Dateinamen innerhalb des Archivs wie auch der Archivname selbst müssen den in den Scheinbedingungen festgelegten Namenskonventionen folgen. Abgaben, die diese Konventionen nicht einhalten, gelten als nicht erfolgt.

Eure Ansätze und der gewählte Lösungsweg müssen nachvollziehbar sein. Achtet insofern auf eine saubere Dokumentation im Quelltext. Benennt alle von euch verwendeten Quellen, auch Zusammenarbeit mit anderen Gruppen und verwendete Unterlagen aus früheren Jahrgängen.

Für Programmieraufgaben ist die Korrektheit der Lösung bzw. deren Grenzen grundsätzlich nachzuweisen. Dies geschieht neben der Dokumentation des Programmcodes durch geeignete Tests, deren Auswahl und Eignung begründet werden muss.