3.1 Virtual memory

(a) Nennen Sie mindestens 2 Vorteile von virtuellem Speicher aus der Sicht eines Betriebssystems oder Anwendungsprogramms.

(b) Das Binärprogramm des Chrome-Browsers ist komplex und hat eine beachtliche Dateigröße. So hat die Bibliothek des Windows-Kompilats von Chrome (`chrome.dll`) etwa eine Dateigröße von 32 MB. Aus Sicherheitsgründen wird jedes einzelne Tab (und jedes Fenster) als separater Prozess geführt. Gegeben sei ein System mit 512 MB physikalischem Speicher und keinem Pagefile. Wie viele Chrome-Tabs können Sie öffnen (ohne eine Webseite zu laden), bevor dem Betriebssystem der Hauptspeicher ausgeht?

(c) In der Vorlesung wurden zwei Paging-Schemata für Pages mit 4 KB Größe besprochen, einmal das reguläre Paging auf 32-bit x86 CPUs (IA-32) sowie Paging mit Physical Address Extension (PAE) unter 32-bit x86 CPUs. Wie sieht das Paging fuer 4 KB Pages auf einer 64-bit CPUs aus (IA-32e)? Erarbeiten Sie ein Diagramm, das dem in der Vorlesung gezeigten ähnelt. Wie viele Ebenen hat das Paging unter x86-64 (IA-32e)? Nehmen Sie die virtuelle Adresse 0x80ff80ee6b als Beispiel und rechnen sie sie (so weit wie möglich) in eine physikalische Adresse um.

3.2 Linux

(a) Nennen Sie typische Privilegien des Superusers (root-Account) unter Linux.

(b) Ein Prozess ist immer einer User-ID zugeordnet. Neben dieser auch als real user ID bezeichneten ID gibt es auch die sog. effective user ID. Was ist der Unterschied zwischen den beiden und gibt es Situationen, in denen man beide benötigt?

3.3 Executable file formats: PE

Die folgenden Aufgaben sollen das Verständnis von Dateiformaten für ausführbare Dateien vertiefen. Manchmal gibt es hierbei mehrere Lösungsmöglichkeiten. Nennen Sie mindestens eine.

(a) Vergegenwärtigen Sie sich die Windows-Datenstruktur `IMAGE_SECTION_HEADER`, die einen Eintrag in der Section Table beschreibt. Ist es denkbar, eine Section Table zu erzeugen, bei der sich zwei Sections im Speicher überlappen? Wenn ja, wie sehen die relevanten Datenstrukturen aus?

3.4 Executable file formats: ELF

Vergegenwärtigen Sie sich den Unterschied zwischen der `program header table` und der `sections table` eines ELF-Executables. Fertigen Sie dazu eine Kopie des Binärprogramms `/bin/ls` der SREVM an. Entfernen Sie nun die `section header table` und entfernen/überschreiben Sie alle Pointer, die aus dem ELF-Header auf die `section header table` oder `sections` zeigen.

(a) Welche Felder im ELF-Header müssen Sie mindestens anpassen?

(b) Ist das modifizierte Programm noch lauffähig? Begründen Sie.

3.5 System calls

(a) In der Vorlesung wurde das Programm `strace` gezeigt, mit Hilfe dessen system calls unter Linux protokolliert werden koennen. Auf welche Funktionalitaet greift `strace` zurueck? Tipp: Versuchen Sie `strace` auf sich selbst anzuwenden.

(b) Optional: Unter `https://blog.nelhage.com/2010/08/write-yourself-an-strace-in-70-lines-of-code/` ist eine exemplarische Beschreibung einer minimalen `strace`-Implementierung. Versuchen Sie, diese Implementierung nachzuvollziehen.