Vorlesung Betriebssysteme WS 2018/2019

Aufgabenblatt 2 vom 20. November 2018

(Vorstellung der Lösungen bei den Tutoren bis zum 04.12.2018)

1. Beantworten Sie folgende Fragen und erläutern Sie die Antworten Ihrem Tutor.
   (a) Nennen und beschreiben Sie kurz vier Designziele beim Entwurf von Betriebssystemen.
   (b) Für welche Schichten der Speicherhierarchie stellt das Betriebssystem Schnittstellen bereit? Nennen Sie je ein Beispiel.
   (c) Was ist ein Kontextwechsel? Beschreiben Sie den Ablauf.

2. Kernel-Objekte repräsentieren Windows-Ressourcen, die von Applikationen verwendet werden können. Beantworten Sie die folgenden Fragen dazu:
   (a) Erläutern Sie für die Objekte *File*, *Mutant*, *Directory* und *Event* welche Betriebsmittel diese repräsentieren und welche Funktionalität sich dahinter verbirgt.
   (b) Was ist ein *Handle*? Wieso können Sie Handles nicht ohne weiteres über Prozessgrenzen hinaus verwenden? Welche Funktionen der Windows API erlauben Ihnen, dies doch zu tun?
   (c) Der Kernel-Adressraum ist in zwei große Speicherbereiche unterteilt. Benennen Sie diese und erläutern Sie den Unterschied. Durch den Verbrauch welchen Speicherbereiches kann das System zum Quasi-Stillstand gebracht werden?
   (d) Untersuchen Sie die Datei `handle.c` des Windows Research Kernels. Ermitteln Sie durch Code-Review die maximale Anzahl an Betriebsmitteln, die ein Prozess besitzen kann. Welche Konstante enthält diesen Wert?
   (e) Nennen Sie fünf Aufgaben des Moduls *Object Manager*.

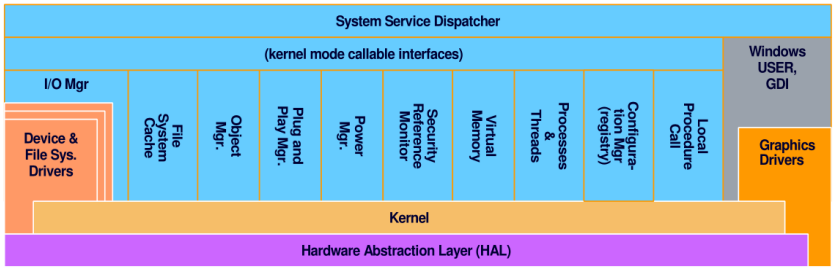

Abbildung 1: Übersicht über die Architektur des Windows-Kernels

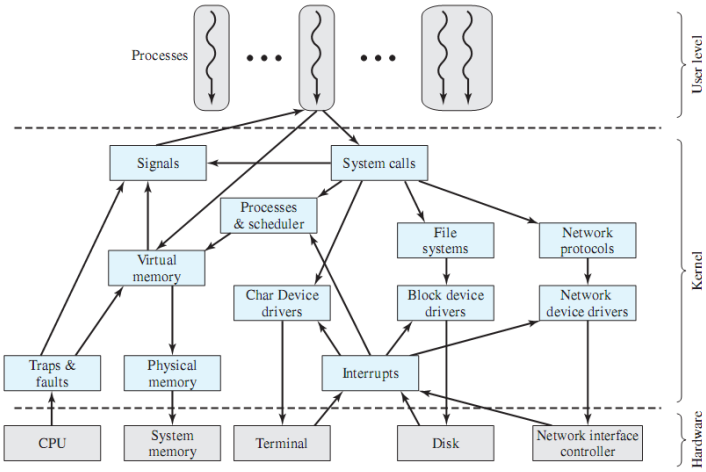

Abbildung 2: Übersicht über die Architektur des Linux-Kernels

3. Die Abbildungen 1 und 2 zeigen den in der Vorlesung vorgestellten Aufbau des Windows- und des Linux-Kernels. Machen Sie sich mit der Dokumentation und den Quellen des Windows Research Kernels (WRK)[1] und des Linux-Kernels[2] in der Version `4.*` vertraut und beantworten Sie die folgenden Fragen:
   (a) Betrachten Sie die Verzeichnisstruktur der Kernel Quellcode Dateien. Beschreiben Sie kurz, welche Teile des Kernels sich in welchen Unterverzeichnissen befinden.
   (b) Der WRK enthält nur einen Teil der Windows Kernel Quellen. Welche Teile fehlen?
   (c) In welchen WRK-Dateien werden Scheduling-Entscheidungen getroffen?

[1] `http://wrk.dcl.hpi.uni-potsdam.de` (erreichbar aus dem HPI-Netz)
[2] `https://www.kernel.org`

4. Setzen Sie sich mit historischen Betriebssystemen auseinander. Untersuchen Sie dazu in der Betriebssystem Galerie[3] mindestens drei der Systeme *Plan 9*, *OS/2 Warp*, *System 6*, *Minix*, *Xenix* oder *Windows 95* in Hinblick auf die folgenden Fragestellungen und erläutern Sie die Erkenntnisse Ihrem Tutor.

   Es kann bei der Bedienung der Betriebssysteme in der Betriebssystem Galerie zu Problemen mit den Tastaturlayouts kommen. Um diese zu umgehen, stellen Sie das Tastaturlayout Ihres Systems auf eine Amerikanische Tastatur um, bevor Sie die Virtuellen Maschinen starten.

   Beachten Sie, dass wir eine Erreichbarkeit der VM Instanzen nur aus dem internen HPI Netz sicherstellen können.

   Sollten Sie dennoch Probleme in der Bedienung feststellen, schreiben Sie uns eine E-Mail an `bs@hpi.uni-potsdam.de`.

   (a) Recherchieren Sie, ob das System über einen *präemptiven* oder einen *kooperativen* Scheduler verfügt.

   (b) Führen Sie auf dem System eine Standardanwendung aus (z.B. einen Editor oder den Taschenrechner). Beschreiben Sie kurz, wie die Interaktion mit der historischen Applikation sich von vergleichbaren Applikationen moderner Systeme unterscheidet.

   (c) Wie können Sie *nicht mehr reagierende* Anwendung mit systemeigenen Werkzeugen beenden? Beschreiben Sie, wie Sie Ihre unter (b) gestartete Anwendung mit einem dieser Werkzeuge beendet haben und machen Sie ein Bildschirmfoto zur Demonstration.

   (d) Versuchen Sie (programmatisch) das System durch Überlastung zum Stillstand zu bringen (z.B. maximale Anzahl Prozesse/Dateien/Verzeichnisse). Dokumentieren Sie Ihr Vorgehen und präsentieren Sie es Ihrem Tutor.

   Stellen Sie Ihre Antworten und Screenshots als PDF mindestens 24 Stunden vor dem Tutoriumstermin in das Abgabesystem ein.

[3] `http://osgallery.instantlab.org`

5. Schreiben Sie basierend auf dem zur Verfügung gestellten Programmrahmen ein C-Programm, das mit Inline-Assembler „manuell“ einen *Systemaufruf* ausführt, um die Prozess-ID des laufenden Prozesses zu ermitteln. Vergleichen Sie das Ergebnis mit der entsprechenden User-Mode API Funktion.

   Ihr Programm soll auf folgenden Systemen funktionieren:

   - Windows 2003 Server Service Pack 1, x86 (32 bit)
   - Linux 3.2.* *(oder neuer)*, x86_64 (64 bit)

   Machen Sie sich dazu mit dem *C-Präprozessor*, sowie den Präprozessorbefehlen `#if` und `#ifdef` vertraut.

   Auf Linux finden sie im Makro `__NR_getpid`[4] die Nummer des Systemaufrufs. Unter Windows 2003 nutzen Sie die Nummer für `NtQueryInformationProcess`[5].

   Sie können die von uns bereitgestellten virtuellen Maschinen[6] verwenden.

   Beantworten Sie zudem folgende Fragen:

   (a) Was ist der Unterschied zwischen `NtQueryInformationProcess` und `GetCurrentProcessId` in Windows?

   (b) Welche strukturellen Unterschiede gibt es in den Systemrufschnittstellen von Windows und Linux?

   (c) Wo lassen sich die Informationen, die über den Windows-Systemruf verfügbar, sind unter Linux wiederfinden?

   (d) Warum ist es eine schlechte Idee Systemaufrufe wie in dieser Aufgabe zu programmieren?

   Stellen Sie ein ZIP-Archiv Ihres Quellcodes mindestens 24 Stunden vor dem Tutoriumstermin in das Abgabesystem ein.

6. **(Zusatz)** Implementieren Sie die letzte Aufgabe unter Linux (x86) und Windows 10 (beliebige Architektur).

[4] Auf Ihrem System vermutlich `/usr/include/asm/unistd.h`
[5] `https://j00ru.vexillium.org/syscalls/nt/32/`
[6] `http://bs1.instantlab.org`