Vorlesung Betriebssysteme WS 2018/2019

Aufgabenblatt 1 vom 30. Oktober 2018

(Vorstellung der Lösungen bei den Tutoren bis zum 12.11.2018)

1. Beantworten Sie folgende Fragen und erläutern Sie die Antworten Ihrem Tutor.
   (a) Was sind die Aufgaben eines Betriebssystems?
   (b) Was ist ein Prozess? Was ist ein Thread? Was ist ein Programm? Erklären Sie die Begriffe und deren Beziehungen zueinander.
   (c) Was sind Vor- und Nachteile der Trennung von Kernel- und Usermode?
   (d) Nennen Sie Beispiele für Betriebssystemfunktionalitäten, die im Kernel- bzw. im Usermode implementiert werden müssen!
   (e) Vergleichen Sie die Konzepte *Mikrokern* und *Monolithischer Kern*. Nennen Sie jeweils ein Beispiel. In welcher Kategorie finden Sie Windows wieder?

2. Demonstrieren und erläutern Sie Ihrem Tutor das unterschiedliche Verhalten von Programmen in Kernel- und Usermode, wenn Sie:
   (a) Eine privilegierte CPU-Instruktion ausführen (z.B. `rdmsr` auf x86_64)
   (b) Einen Programmierfehler machen (z.B. ungültiger Zeigerzugriff)

   Stellen Sie Ihren Quelltext bis 24 Stunden vor dem Tutoriumstermin ins Abgabesystem ein. Hinweise:

   - Sie können einen vorbereiteten Programmrahmen[1] verwenden, der ein Makefile und Gerüst für ein einfaches Linux-Kernelmodul enthält.
   - Die Ausgabe von `printk` im Linuxkernel kann mithilfe des Kommandozeilenbefehls `dmesg` angezeigt werden.
   - Moderne Linux-Systeme versuchen einfache Fehler (*Oops*) abzufangen. Sie können dies (temporär) deaktivieren:

     ```
     sudo sh -c 'echo 1 > /proc/sys/kernel/panic_on_oops'
     ```

[1] `https://www.dcl.hpi.uni-potsdam.de/teaching/BS/HA/blatt1/rdmsr.tar.gz`

3. Betrachten Sie Dateien auf einem Unix-änlichen System, wie zum Beispiel GNU/Linux oder MacOS, und untersuchen Sie deren Zugriffsberechtigungen. Beantworten Sie dazu die folgenden Fragen Ihrem Tutor.
   (a) Was ist die Bedeutung von `rwx`-Bits? Welche Programme erlauben Ihnen, die `rwx`-Bits einer Datei oder eines Verzeichnisses einzusehen oder zu ändern?
   (b) Welche weiteren Bits gibt es und was ist deren Semantik?
   (c) Beschreiben Sie im Speziellen das *setuid*-Bit. Welche Funktionalität verbirgt sich dahinter? Welche Programme nutzen diese und warum?
   (d) Diskutieren Sie mögliche Sicherheitsprobleme, die mit der Benutzung des *setuid*-Bits einher gehen. Welche Alternativen zum *setuid*-Bit existieren?

4. Schreiben Sie ein C-Programm unter Linux oder macOS, welches eine *EBCDIC*-kodierte Textdatei nach *ASCII* konvertiert.

   Ihr Programm soll den Pfad zu einer EBCDIC-kodierten Textdatei als Kommandozeilenparameter erwarten. Diese Datei soll nach ASCII konvertiert und auf der Standardausgabe ausgegeben werden. Beachten Sie dabei insbesondere, dass EBCDIC-kodierte Dateien traditionell nur einen *Carriage Return* als Zeilenende verwenden, und keine *Line Feed*-Zeichen enthalten. Wie gehen Sie in Ihrer Ausgabe damit um? Erläutern Sie Ihre Gedanken dazu Ihrem Tutor.

   Achten Sie in Ihrer Implementierung außerdem auf die korrekte Behandlung von Laufzeit- und Eingabefehlern. Geben Sie im Fehlerfall sinnvolle Fehlermeldungen aus, zum Beispiel durch Verwendung der Funktionen `perror` oder `strerror`.

   Verpacken Sie den Quellcode (inklusive Makefile) in ein Tar- oder Zip-Archiv. Dieses Archiv muss 24 Stunden vor dem Tutoriumstermin in das Abgabesystem eingestellt werden. Demonstrieren Sie Ihr Programm Ihrem Tutor.