Vorlesung Betriebssystemarchitektur WS 2002/03

Aufgabenblatt 4 vom 4. Dezember 2002

(Vorstellung der Lösungen bei den Tutoren bis zum 20. Dezember 2002)

Aufgabe 4.1: (10 Punkte)
Erweitern Sie das Shell-Programm von Aufgabenblatt 2 um die Möglichkeit, das aktuelle Verzeichnis zu ändern (**`cd`**). Außerdem soll als Eingabe-Aufforderung das aktuelle Verzeichnis ausgegeben werden. Sie können sich aussuchen, ob Sie die Linux- oder Windows-Variante bearbeiten möchten. Wenn bei **`cd`** kein Parameter angegeben ist, dann soll die Funktion **`go_to_temp_dir()`** aufgerufen werden. Diese Funktion befindet sich in der Datei **`go_to_temp_dir.c`**, die auf der Webseite **`http://www.dcl.hpi.uni-potsdam.de/uebung2002/blatt4`** zu finden ist. Diese Funktion wechselt in ein temporäres Verzeichnis, das gegebenenfalls noch angelegt wird. Beispiel:

```
$ minishell
/home/stud/m/mdirska > cd test
/home/stud/m/mdirska/test > /bin/echo Hallo
Hallo
Rueckgabewert: 0
/home/stud/m/mdirska/test > cd
/tmp/mdirska > ^D
$
```

Führen Sie Ihrem Tutor das Programm vor und erklären Sie die Änderungen.

Aufgabe 4.2: (20 Punkte)
Betätigen Sie sich als Hacker. Ändern Sie Ihre Entwicklungsumgebung so ab, dass Ihr Programm aus Aufgabe 4.1, wenn Sie es neu übersetzen (mit identischen Compiler-Aufrufen, ungeänderte Quell-Dateien) beim Aufruf der Funktion **`go_to_temp_dir()`** etwas völlig anderes tut, z.B., den Inhalt des temporären Verzeichnisses löscht. Erklären Sie Ihrem Tutor, was Sie dazu ändern mussten. Wie lässt sich die von Ihnen ausgenutzte Sicherheitslücke schließen?

Aufgabe 4.3: (30 Punkte)
In einer Firma sind 100 Angestellte beschäftigt, die über eine bestimmte Menge an identischen Standard-Zugriffsrechten verfügen. Desweiteren existieren 2 privilegierte Nutzer, die über zusätzliche Rechte verfügen sollen (z.B. Administratoren).

Es soll ein gemeinsamer Ordner für die Angestellten erzeugt werden, auf den die beiden privilegierten Nutzer jedoch keinen Zugriff haben sollen.

- Wie kann das beschriebene Szenario mit den Betriebssystemmitteln unter UNIX (rwx-Bits, Gruppen) umgesetzt werden?
- Wie mit Hilfe von Windows 2000 ?

Nach einer Firmenzusammenlegung sollen 15 zusätzliche Nutzer mit Standardrechten angelegt werden, die zunächst jedoch keinen Zugriff auf den oben beschriebenen gemeinsamen Ordner haben dürfen.

- Wie kann dieses Ziel mit UNIX erreicht werden ?
- Wie mit Windows 2000 ?

Schreiben Sie Ihre Lösungsvorschläge in Stichpunkten auf und fertigen Sie eine Skizze an. Geben Sie die Ergebnisse zum Tutoriums-Termin in Papierform ab.

Aufgabe 4.4: (40 Punkte)
Auf den Linux-Rechnern in den Studenten-Übungsräumen im Haus C ist ein Modul namens "uebung" in den Betriebssystem-Kern geladen. Dieses Modul implementiert einen zeichenorientierten Gerätetreiber, der über den Dateinamen **`/dev/uebung`** angesprochen werden kann. Der Quell-Code zu diesem Modul befindet sich auf obengenannter Webseite. Der Gerätetreiber implementiert einen FIFO-Speicher, dessen Speichertiefe von außen per ioctl(2)-Systemaufruf inkrementiert werden kann (siehe Beispiel-Programm ioctl.c auf der Webseite). Mit write(2) können Daten in den FIFO-Speicher geladen werden, wenn dort Platz vorhanden ist. Mit read(2) können zuvor geschriebene Daten aus dem FIFO-Speicher gelesen werden. Die Systemaufrufe blockieren den jeweiligen Prozess, wenn entweder kein Speicher zum Schreiben verfügbar ist bzw. keine Daten zum Lesen vorhanden sind. Der Gerätetreiber enthält einige printk-Aufrufe, dessen Ausgabe in der Datei **`/var/log/debug`** gespeichert wird.

Machen Sie sich mit dem Befehl strace(1) vertraut. Schreiben Sie kleine Test-Programme in C, mit denen Sie die Funktionsweise des Gerätetreibers herausfinden können. Beantworten Sie Ihrem Tutor folgende Fragen:

- Erläutern Sie grob die Funktionsweise des Gerätetreibers. Was bewirkt der Aufruf von ioctl?
- Was passiert, wenn mit einem write(2)-Systemaufruf mehr Daten geschrieben werden sollen, als gerade Platz im FIFO-Speicher vorhanden ist?
- Was passiert, wenn Sie mit einem read(2)-Systemaufruf aus dem FIFO-Speicher mehr Daten lesen wollen als dort zum Abruf bereitstehen?
- Was passiert, wenn mehrere Prozesse gleichzeitig aus dem FIFO-Speicher lesen wollen? Bekommen alle Prozesse dieselben Daten oder bekommt immer nur einer etwas zurück?
- Informieren Sie sich über den Unterschied zwischen den Systemaufrufen open(2), read(2), write(2) und den "stream"-Funktionen fopen(3), fread(3), fwrite(3). Wo ist der Unterschied?
- Was ist der Unterschied zwischen stdout (aus der Header-Datei stdio.h) und dem File-Deskriptor 1?
- Kompilieren Sie das C-Programm buffertest.c (auf der Webseite). Starten Sie das Programm auf der Konsole. Ermitteln Sie mit strace, wieviele write-Systemaufrufe dabei getätigt werden.
- Lenken Sie die Ausgabe von buffertest in den FIFO-Speicher um. Wieviele write-Systemaufrufe erzeugt das Programm jetzt? Gibt es Unterschiede zur Konsolenausgabe? Wenn ja, überlegen Sie sich, woran das liegen könnte.
- Ermitteln Sie die Ein- und Ausgabe-Buffergröße der Befehle cat(1) und dd(1) mit Hilfe von strace und dem FIFO-Speicher. Erklären Sie Ihrem Tutor, wie Sie dabei vorgegangen sind.