Übungen zur Vorlesung
Betriebssysteme II
Sommersemester 2002

AG BS
Jan Peleska, Aliki Tsiolakis

Blatt 1

Linux Kernel – Neue System-Calls und Kernel-Generierung

Aufgabe 1: **20%**

Erarbeiten Sie eine Liste der Schritte, welche zur Erstellung eines neuen Linux Kernels erforderlich sind, wenn **nur Kernel Sources** geändert wurden und daher alle existierenden Makefiles aus `/usr/src/linux` weiter verwendet werden können. Die Schritte sollen die Verwendung des LILO Loaders beinhalten, so dass der neu generierte Kernel beim Boot alternativ zum existierenden angwählt werden kann.

Aufgabe 2: **20%**

Erarbeiten Sie eine Liste der Schritte, welche zur Erzeugung eines neuen Systemaufrufs erforderlich sind. Dabei soll vorausgesetzt werden, dass die Implementierung des Aufrufs in einer der bereits existierenden Quelldateien des Kernels erfolgen kann (siehe nächste Aufgabe). Es müssen also keine neuen Quelldateien oder Makefiles oder Module erzeugt werden.

Aufgabe 3: **60%**

Erzeugen Sie wahlweise einen der beiden neuen Systemaufrufe

1. `pid_t my_clone(int (*fn) (void *arg), void *child_stack, void *arg)`

   Dieser Call soll dieselbe Wirkungsweise wie `clone(2)` mit den üblicherweise benutzten Flages `CLONE_VM`, `CLONE_FS`, `CLONE_FILES`, `CLONE_SIGHAND` besitzen.

   VORSICHT: Hier muss man einen Wrapper analog zu `sys_clone()` in `process.c` bauen, der berücksichtigt, dass jetzt die *clone flags* NICHT mehr als User Parameter in `regs.ebx` stehen.

2. `pid_t my_fork()`

   Dieser Call soll dieselbe Wirkungsweise wie fork besitzen mit folgendem Unterschied: `do_my_fork()` ist eine Kopie von `do_fork()`, ruft aber `get_my_pid()` anstelle von `get_pid()` auf. `get_my_pid()` vergibt immer die kleinste freie $pid > 300$. Es wird hier als immer die gesamte Prozesstabelle durchsucht, um *pid* zu finden.
   (Diese Aufgabe ist leichter als Aufgabe 3.1.)

Es sollen dieselben Verfahren zur Registrierung des Aufrufs auf Grundlage von `linux/include/asm-i386/unistd.h` verwendet werden. Um nicht auch noch die glibc erweitern zu müssen, darf der Wrapper und das zugehörige Header-File separat erzeugt werden. Implementieren Sie den Kernelcode als neue Funktion in `linux/kernel/fork.c`. Generieren Sie den neuen Kernel und schreiben Sie ein Beispielprogramm, welches die Funktionsfähigkeit des neuen Aufrufs demonstriert.

Abgabe: Bis Montag, 6. Mai 2002, vor dem Tutorium.

Geben Sie für alle Aufgaben eine **schriftliche Lösung** ab. Diese soll für Aufgabe 3 auch die Dokumentation (Lösungsansatz, Vorgehen, Testfälle und -ergebnisse) sowie die veränderten Dateien der Kernel-Sourcen enthalten. Bitte schicken Sie **zusätzlich** ein Archiv[1] aller relevanten Dateien per Email an `tsio@informatik.uni-bremen.de`.

Sowohl bei der schriftlichen Lösung als auch im Source-Code die Namen aller Gruppenmitglieder nicht vergessen!

[1] Für die „Archivierung“ werden alle relevanten Dateien (aber keine Binaries :-) in ein Verzeichnis kopiert und das Verzeichnis dann mit `tar`, `gzip`, ... zusammengefasst. Die Benennung des Verzeichnisses sollte möglichst die Gruppennummer sowie die Nummer des Aufgabenblattes enthalten, z.B. `blatt1-grp25`.