Operating Systems Home Assignment

Aufgabe 2
Prozesse und System Calls

Prof. Dr.-Ing Jens Heuschkel
Stephan Gimbel
Hochschule Darmstadt

WS 2025

Das Praktikum zur Vorlesung Betriebssysteme dient der Vertiefung der in der Vorlesung behandelten Inhalte. Die Aufgaben sollen eigenständig zu Hause bearbeitet werden und sind nicht benotet. Eine Abgabe der Lösungen ist nicht erforderlich. In den Präsenzterminen kann es Folgeaufgaben geben, die auf den zuvor behandelten Übungsinhalten aufbauen.

Lernziele

- Die System Calls `fork`, `execvp`, `waitpid` und die Signalbehandlung kennen und anwenden lernen
- Prozesse im Unix-/Linux-Betriebssystem verstehen und steuern
- Grundlegende Shell-Funktionalitäten wie Hintergrundausführung, Umleitungen und Built-In-Befehle implementieren
- Umgang mit Eingabeparsing und Fehlerbehandlung üben

Aufgabe: Implementierung einer Linux Shell

Implementieren Sie in C (unter Linux) ein minimales Shell-Programm mit folgenden Erweiterungen und Abwandlungen:

1. Lesen Sie in einer Schleife Befehlszeilen ein und zerlegen Sie diese in Programmnamen und Parameter (Tokenizing).
2. Unterstützen Sie Built-In-Befehle[1]:
   - `pwd` gibt das aktuelle Arbeitsverzeichnis aus.
   - `exit` beendet Ihre Shell. Stellen Sie zuvor per Rückfrage sicher, dass der Benutzer wirklich beenden möchte.
3. Starten Sie fremde Programme in einem Kindprozess:
   - Standardmodus: warten Sie auf Beendigung des fremden Programms, bevor Sie den nächsten Prompt anzeigen (Foreground).
   - Mit angehängtem `&`: geben Sie sofort den Prompt zurück und lassen den Prozess im Hintergrund laufen (Background).

   Nach jedem `fork/execvp`-Aufruf geben Sie die PID des gestarteten Prozesses aus.
4. Testen Sie das Verhalten Ihrer Shell mit:
   - einem kurzen Programm (z. B. `ls`), sowohl im Vordergrund als auch im Hintergrund;
   - einem länger laufenden Programm/Editor (z. B. `nano` oder `sleep 30`);
   - Tippfehlern (z. B. `sl` statt `ls`) und deren Fehlermeldungen.

   Dokumentieren Sie Ihre Tests durch Protokollierung der Ausgaben und Beobachtungen.
5. Vergleichen Sie mit `ps ax` (in einer separaten Shell) Ihre ausgegebenen PIDs mit den tatsächlichen Prozessen. Was beobachten Sie bei bereits beendeten Hintergrundprozessen?

Hinweise

- Ihr Programm soll prinzipiell beliebige andere Programme (z. B. auch `firefox`, `gedit`, `ls` …) starten können. Diese Programme sollen nicht hardcodiert sein.
- Der Begriff Vordergrund- bzw. Hintergrundprozess ist dem Betriebssystem nicht bekannt. Das entsprechende Verhalten wird nur durch Ihr Programm implementiert.
- Statt `ps` können Sie auch `pstree` verwenden, um die Prozesshierarchie anzuzeigen.
- Um Prozesse länger zu beobachten, eignet sich auch `top`.

Vorbereitung für die Präsenzübung

- Lesen Sie sich in die Signale `SIGTSTP`, `SIGCONT`, `SIGTERM` und `SIGKILL` ein.
- Was bewirken die jeweiligen Signale?
- Wie können Sie die Signale senden und empfangen?

[1] Built-In-Befehle sind direkt in Ihrer Shell implementiert und rufen kein separates Programm auf.