SPiC-Aufgabe #8: `tbsh`

(12 Punkte, Abgabe bis Montag, 25.07.2011, um 8:00, in Zweier-Gruppen)

Entwickeln Sie auf Basis der `tish` aus der vorherigen Aufgabe nun eine Shell `tbsh` (token bucket shell), die um folgende Funktionalität erweitert werden soll:

1. Ignorieren des INT-Signals

   Erweitern Sie die Shell so, dass das INT-Signal (erzeugt z. B. durch Drücken von CTRL-C) von Ihrer Shell (aber nicht von den erzeugten Kindprozessen!) ignoriert wird (`sigaction(2)`). Sie sollten nun laufende Kindprozesse durch Eingabe von CTRL-C abbrechen können, ohne jedoch dabei Ihre Shell zu beenden.

2. Begrenzung der maximalen Prozesszahl pro Zeiteinheit

   Die Shell soll nun einen Mechanismus implementieren, der die Zahl an Prozessen, die im Zuge einer Kommandoausführung erzeugt werden, begrenzen soll. Hierzu vergibt die Shell in bestimmten Zeitabständen (refresh interval) Token an den Benutzer. Ein Token erlaubt die Ausführung eines Prozesses und wird hierbei verbraucht. Die maximale Zahl an Tokens (`max_tokens`), die ein Benutzer ohne einen Prozess zu starten durch Warten ansparen kann, ist begrenzt. Der Mechanismus soll im Detail wie folgt funktionieren:

   - Die beiden Parameter refresh interval und max tokens können der Shell als Kommandozeilenparameter übergeben werden. Es müssen entweder beide Werte oder keiner der Werte übergeben werden. In letzterem Falle ist die Zahl der Prozesse wie schon in der `tish` nicht begrenzt. Ansonsten steht zum Programmstart *ein* Token zur Verfügung.
   - Wird von dem Mechanismus Gebrauch gemacht, aktiviert die Shell einen Systemalarm (`alarm(2)`), der nach refresh interval Sekunden abläuft und der Shell dann ein ALRM-Signal zustellt.
   - Die von der Shell für das ALRM-Signal installierte Signalbehandlungsroutine (`sigaction(2)`) erhöht die Zahl der verfügbaren Tokens um 1, falls die maximale Zahl max tokens noch nicht erreicht ist.
   - Vor dem Start eines Prozesses wird überprüft, ob noch Token zur Verfügung stehen. Sind derzeit alle Token verbraucht, gibt die Shell eine entsprechende Fehlermeldung aus und führt das Kommando nicht aus. Ansonsten wird das Kommando gestartet und ein Token verbraucht. Achten Sie hierbei auf evtl. bestehende Nebenläufigkeitsprobleme mit Signalbehandlungen und synchronisieren Sie entsprechend (`sigprocmask(2)`).

3. Shell-Prompt

   Zuletzt soll noch das Prompt-Symbol der Shell angepasst werden. Wird vom time-quota-Mechanismus Gebrauch gemacht, so soll die Shell einen Prompt der Form `tbsh[num_tokens]>` ausgeben, wobei an Stelle von `num_tokens` die zum Zeitpunkt der Promptausgabe verfügbare Zahl von Tokens eingesetzt wird. Wird der Mechanismus nicht verwendet, so wird an Stelle der Tokenzahl `unlimited` ausgegeben.

Hinweise:

- Ihr Programm muss mit folgenden Flags warnungs- und fehlerfrei kompilieren:

  ```
  gcc -std=c99 -pedantic -Wall -Werror -D_XOPEN_SOURCE=500 -o tbsh tbsh.c
  ```

- Sie können vereinfachend davon ausgehen, dass die Länge einer Kommandozeile maximal 1023 Zeichen beträgt. In anderen Fällen muss Ihre Shell nicht mehr korrekt funktionieren, es dürfen jedoch keine Pufferüberläufe auftreten.