ulm university universität
uulm

Institut für Numerische Mathematik

Dr. Andreas F. Borchert
Fabian Berstecher

11. Mai 2017
Blatt 4

Systemnahe Software II (SS 2017)

Abgabe bis zum 18. Mai 2017, 16:00 Uhr

Lernziele:

- Aufsetzen eines Signalbehandlers und der Umgang mit Wecksignalen.

Aufgabe 4: Prozess unter Aufsicht

Hans bittet um Ihre Hilfe. Er würde gerne auf dem Server einen rechenintensiven Job starten, aber er weiß noch nicht, wie lange sich der Job hinziehen wird. Wenn der Job länger als 5 Minuten benötigt, muss nach den Spielregeln seine Priorität heruntergesetzt werden. (Er sieht es aber nicht ein, dies vorher zu tun, da dies bei kurzzeitigen Jobs noch nicht notwendig ist.) Eine Veranstaltung, die eine Stunde später angesetzt ist und den Server für eine Stunde benötigt, will er ebenfalls nicht stören. In dieser Zeit soll der Job einfach schlafen gelegt werden. Und sollte der Job danach noch einmal mehr als eine Stunde benötigen, dann ist er zu terminieren, da er sich dann das doch noch einmal ansehen müsse. Falls der Prozess nicht bei *SIGTERM* terminiere, sei dies mit *SIGKILL* fünf Sekunden später zu forcieren.

Sie bieten Ihm darauf an, ein Werkzeug namens *pcontrol* zu entwickeln, das diese Aufgabe löst. Das Kommando nimmt den Job (als beliebiges Kommando) entgegen und zusätzlich beliebig viele weitere Kommandos mit inkrementellen Zeitangaben, die dann auszuführen sind, wenn die jeweilige Zeit erreicht wird und der Job immer noch läuft. Hier wäre dann das Beispiel für Hans:

```
pcontrol / job \
   /  300 renice -n 10 -p % \
   / 3300 kill -STOP % \
   / 3600 kill -CONT % \
   / 3600 kill -TERM % \
   /    5 kill -KILL %
```

Da die einzelnen Kommandos jeweils beliebig viele Argumente haben können, definiert das allererste Argument von *pcontrol* das Trennsymbol. In diesem Beispiel ist das der Schrägstrich „/“, aber dies darf auch eine beliebige andere Zeichenkette sein. Als zweites folgt das

Kommando des zu beaufsichtigenden Prozesses. Danach kommen, getrennt jeweils durch das Trennsymbol, beliebig viele weitere Kommandos jeweils mit einer inkrementellen Zeitangabe in Sekunden. Wenn der Job irgendwann terminiert, terminiert auch *pcontrol*. Wenn der Job jedoch weiterläuft und die nächste Zeitangabe erreicht wird, dann ist das jeweilige Kommando aufzurufen, dessen Erfolg jedoch nicht weiter überprüft oder verfolgt wird.

Wenn eines dieser Kommandos die Prozess-ID des zu beaufsichtigenden Prozesses benötigt, kann „%“ angegeben werden, das von *pcontrol* durch die Prozess-ID ersetzt wird.

Wenn bei *pcontrol* irgendetwas fehlschlägt, hat es mit einem Exit-Wert von 255 zu terminieren. Wenn der zu beaufsichtigende Prozess terminiert, muss *pcontrol* den Exit-Code, sofern existent, übernehmen. Andernfalls ist ein Exit-Code von 255 zu verwenden.

Hinweise:

Die Systemaufrufe *wait* und *waidpid* kennen kein Zeitlimit. Zwar unterstützt *waitpid* die Option *WNOHANG*, aber das hilft hier nicht weiter, da der *pcontrol*-Prozess kontinuierlich schlafen soll, bis entweder der zu beaufsichtigende Prozess terminiert oder das nächste Timeout-Kommando zur Ausführung fällig ist.

Deswegen bietet es sich an, *wait* oder *waitpid* mit *alarm* zu kombinieren und auf das Eintreten von *SIGALRM* zu reagieren. *alarm* kann dann jeweils mit den angegebenen Timeout-Werten aufgerufen werden. (Da diese bereits inkrementell sind, können sie direkt übernommen werden.)

Da die Zahl der Timeout-Kommandos beliebig ist (aber mindestens eines vorhanden sein muss), bietet sich eine lineare, einfach verkettete Liste als geeignete dynamische Datenstruktur an.

Anders als bei der *tinysh* aus der Vorlesung ist es nicht notwendig, die Argumentlisten neu zu erzeugen. Stattdessen reicht es völlig aus, die Zeiger auf die Trennsymbole durch Nullzeiger zu ersetzen und die Zeiger auf die „%“ durch Zeiger auf eine Zeichenkette mit der Prozess-ID zu ersetzen. Dann können Zeiger auf die entsprechenden Teilbereiche der ursprünglichen *argv*-Argumentkette an *execvp* übergeben werden.
Ihre Lösung können Sie dann mit *submit* einreichen:

```
thales$ submit ss2 4 pcontrol.c team
```

Viel Erfolg!