Aufgabe 4: clash (12 Punkte) Bearbeitung in Zweier-Gruppen

18.11.2009

Implementieren Sie ein Programm clash (**C l**anguage **a**pprentice's **sh**ell) in einer Datei **clash.c**, das Programme (im Weiteren als Kommandos bezeichnet) ausführt.

clash gibt als Promptsymbol das aktuelle Arbeitsverzeichnis (**getcwd(3)**) gefolgt von einem ":"-Zeichen aus und liest dann eine Zeile von der Standardeingabe ein. Die eingelesene Zeile wird in Kommandonamen und Argumente zerlegt, als Trennzeichen dienen Leerzeichen und Tabulatoren (**strtok(3)**). Das Kommando wird dann in einem neu erzeugten Prozess (**fork(2)**) mit korrekt übergebenen Argumenten ausgeführt (**exec(2)**).

a) Vordergrundprozesse

clash wartet anschließend auf das Terminieren der Kommandoausführung (**waitpid(2)**) und gibt den Exitstatus aus. Bei der Ausgabe soll unterschieden werden, ob der Prozess sich aktiv selbst beendet hat, oder ob der Prozess durch ein Signal beendet wurde. Die Ausgabe soll wie folgt aussehen:

1. Fall: Prozess beendet sich selbst:

```
/proj/i4sp/mike: ls -l
...
Exitstatus [ls -l] = 0
```

2. Fall: Prozess wird durch ein Signal beendet:

```
/proj/i4sp/mike: ./sp_wait
...
Signal [./sp_wait] = 2
```

Nach der Ausgabe des Exitstatus nimmt die Shell wieder eine neue Eingabe entgegen. clash terminiert bei Signalisierung von End-of-File (CTRL-D) auf dem Standardeingabekanal.

b) Hintergrundprozesse

Endet eine Kommandozeile mit dem *Token* '&', so wird das Kommando in einem Hintergrundprozess ausgeführt. In diesem Fall wartet die Shell nicht auf die Beendigung des Prozesses, sondern zeigt sofort einen neuen Prompt zur Entgegennahme weiterer Kommandos an. Jeweils vor Anzeige eines neuen Prompts sammelt die Shell alle zu diesem Zeitpunkt terminierten Hintergrundprozesse (Zombies) auf und gibt deren Exitstatus analog zu den Vordergrundprozessen aus. Merken Sie sich hierfür beim Erzeugen eines Hintergrundprozesses dessen PID und Kommandozeile in einer verketteten Liste, die in einem Modul **plist.c** die im vorgegebenen Header */proj/i4sp/pub/aufgabe4/plist.h* beschriebene Schnittstelle implementiert. Sie können ggf. Ihre **slist** aus Aufgabe 1 anpassen und nicht mehr benötigte Funktionen entfernen.

c) Verzeichniswechsel

Implementieren Sie nun noch einen Verzeichniswechsel (**chdir(2)**). Wird als Kommando "cd" eingegeben, so soll der clash-Prozess sein Arbeitsverzeichnis auf den im nachfolgenden Argument angegebenen Pfad setzen.

d) Makefile

Erstellen Sie ein zur Aufgabe passendes Makefile, welches keine ins **make(1)**-Programm eingebauten Makros und Regeln voraussetzt. Das Makefile sollte mit dem Aufruf *make -Rr* funktionieren.

Hinweise

- In */proj/i4sp/pub/aufgabe4* finden Sie die Programme *clash* und *sp_wait* zum Vergleichen bzw. Testen.
- Die Modulschnittstelle von **plist** ist vorgeschrieben und darf nicht verändert werden. Hilfsfunktionen dürfen nicht Teil der Schnittstelle werden (*static*). Ihre Implementierung sollte auch mit unserer Implementierung von **plist** funktionieren, welche wir in */proj/i4sp/pub/aufgabe4/plist.o* zum Testen bereitstellen. Eine Online-Beschreibung der Schnittstelle ist neben der Aufgabenstellung über die Webseite abrufbar.
- Im Vorgabeverzeichnis findet sich ein Testfall für das **plist**-Modul. Zur Verwendung rufen Sie aus Ihrem **src**-Verzeichnis, das die Datei **plist.c** enthält, das Testskript */proj/i4sp/pub/aufgabe4/runtest.sh* auf.

Abgabe: bis spätestens Dienstag, 1.12.2009, 17:30 Uhr