Aufgabe 4: clash (12.0 Punkte)

Implementieren Sie ein Programm `clash` (**C l**anguage **a**pprentice's **sh**ell), das Programme (im Weiteren als Kommandos bezeichnet) ausführt. `clash` gibt als Promptsymbol das aktuelle Arbeitsverzeichnis (**getcwd(3)**) gefolgt von einem Doppelpunkt 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(3)**).

a) Vordergrundprozesse

`clash` wartet anschließend auf das Terminieren der Kommandoausführung (**waitpid(2)**) und gibt den Exitstatus aus. Die Ausgabe soll wie folgt aussehen:

```
/proj/i4sp1/schedel: ls -l
...
Exitstatus [ls -l] = 0
```

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 bis 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 Implementierung dieser verketteten Liste ist im Modul `plist.c` im Verzeichnis `/proj/i4sp1/pub/aufgabe4/plist.c` vorgegeben.

c) 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 und die Standardtargets `all` und `clean` mit der üblichen Funktionalität bereitstellen.

d) 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.

e) Anzeige laufender Hintergrundprozesse

Implementieren Sie ein Kommando `jobs`, welches in jeweils einer Zeile PID und Kommandozeile aller aktuell laufenden Hintergrundprozesse auf die Standardausgabe ausgibt. Dazu ist es notwendig, das vorgegebene `plist.c`-Modul um die Funktion `walkList()` zu erweitern, die für jedes Element der verketteten Liste eine Callback-Funktion aufruft.

Hinweise zur Aufgabe:

- Erforderliche Dateien: `clash.c, plist.c, Makefile`
- In `/proj/i4sp1/pub/aufgabe4` finden Sie das Programm `clash` zum Vergleichen bzw. Testen.
- Die Modulschnittstelle der *plist* ist vorgeschrieben und darf nicht verändert werden. Hilfsfunktionen dürfen nicht Teil der Schnittstelle werden ($\rightarrow$ `static`). Ihre Shell sollte auch mit unserer Implementierung der *plist* funktionieren, welche wir in `/proj/i4sp1/pub/aufgabe4/plist.o` zum Testen bereitstellen. Eine Online-Beschreibung der Schnittstelle ist neben der Aufgabenstellung über die Webseite abrufbar.
- Der **sysconf(3)**-Parameter `_SC_LINE_MAX` gibt die maximale Länge einer Eingabezeile an, die Ihre `clash` verarbeiten können soll. Sollten mehr Zeichen eingeben werden, soll die gesamte (überlange) Zeile mit einer Warnung verworfen werden.

Hinweise zur Abgabe:

Bearbeitung: Zweiergruppen
Bearbeitungszeit: 10 Werktage
Abgabezeit: 17:30 Uhr