Wichtig: Lesen Sie auch den Teil "**Hinweise zur Aufgabe**" auf diesem Blatt; Spezifikationen in diesem Teil sind ebenfalls einzuhalten!

Aufgabe 3: clash (14.0 Punkte)

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

a) Makefile

Erstellen Sie ein zur Aufgabe passendes Makefile, welches die Targets `all`, `clean` und `clash` unterstützt. Greifen Sie dabei stets auf Zwischenprodukte (z. B. `plist.o`) zurück. Das Makefile soll ohne eingebaute Regeln funktionieren (**make(1)** mit den Optionen `-rR` starten). Nutzen Sie die auf der Webseite genannten Compilerflags.

b) Basisfunktionalität

`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)**).
`clash` wartet bei Vordergrundprozessen auf das Terminieren der Kommandoausführung (**waitpid(2)**) und gibt den Exitstatus gemeinsam mit der zugehörigen Befehlszeile aus. Die Ausgabe soll wie folgt aussehen:

```
/proj/i4sp1/pub/aufgabe3: echo test
test
Exitstatus [echo test] = 0
```

Nach der Ausgabe des Exitstatus nimmt die Shell wieder eine neue Eingabe entgegen.

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` in Ihrem durch `mkrepo` erstellten Git-Verzeichnis vorgegeben.
`clash` terminiert bei Signalisierung von End-of-File (Ctrl-D) auf dem Standardeingabekanal.

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) 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.

Weitere Spezifikationen und Hinweise zur Aufgabe:

- Erforderliche Dateien: `clash.c` (10 Punkte), `plist.c` (2 Punkte), `Makefile` (2 Punkte)
- Im durch `mkrepo` erstellten Git-Repository finden Sie im `pub/` Ordner Vorgabedateien zum Vergleichen bzw. Testen.
- Die Modulschnittstelle der *plist* ist vorgeschrieben und darf nicht verändert werden. Hilfsfunktionen dürfen nicht Teil der Schnittstelle werden (→ `static`). Ihre Shell sollte auch mit der gegeben Implementierung der *plist* funktionieren.
- Eine Beschreibung der *plist*-Schnittstelle ist ebenfalls in Ihrem Repository und kann mit jedem Web-Browser geöffnet werden (`doc/index.html`).
- Der Aufruf der Funktion **getcwd(3)** mit `NULL` als erstem Parameter (z.B. `getcwd(NULL, 0);`) ist **nicht** Teil der POSIX-Spezifikation und **darf somit nicht verwendet werden**. Eine Beschreibung des richtigen Vorgehens zur Allokation eines ausreichend großen Puffers ist in der Manpage im Abschnitt **Description** beschrieben.
- Die maximale Länge einer Eingabezeile (inklusive `\n`), die Ihre `clash` verarbeiten können soll, ist 1337 Bytes. Sollten mehr Zeichen eingeben werden, soll die gesamte (überlange) Zeile mit einer Warnung verworfen werden.
- Fehlerbehandlung für die Ausgabe ist bei dieser Aufgabe optional, da die Ausgabe einer interaktiven Shell nicht auf die Platte geschrieben wird.

Hinweise zur Abgabe:

Bearbeitung: Dreiergruppen
Bearbeitungszeit: 10 Werktage (ohne Wochenenden und Feiertage)
Abgabezeit: 17:30 Uhr