Dr. M. Grabert
Johannes Mayer

Systemnahe Software (SS 2001)

Abgabetermin: 5. Juni 2001

8 Let it grow – Midi-Shell wird erwachsen

Sie haben mal so zum Spass die Funktionalität der Bash-Shell mit der Funktionalität der *Midi-Shell* aus der Vorlesung verglichen und sind ganz enttäuscht. „So winzig und nennt sich schon Shell", denken Sie. Da gibt's nur Eines. Sie kennen ja jetzt einige Funktionalitäten der Bash, von denen Sie ganz begeistert sind. Diese wollen Sie unbedingt in die Midi-Shell aus der Vorlesung einbauen. Gesagt getan! Sie setzen sich an den Rechner und erweitern die Midi-Shell um die folgenden Fähigkeiten.

8.1 Zombie-Vermeidung (2 Punkte)

Wenn sie in der Midi-Shell bisher ein Hintergrundkommando starten, geistert nach der Termination des zugehörigen Prozessen ein Zombie herum. Sie beschließen, dass es durch Ihre Erweiterung erst gar nicht zu Zombies kommen wird.

8.2 Tilde-Expansion (1 Punkt)

Damit sie nicht immer wieder Ihren Heimatkatalog eintippen müssen, ersetzen sie alle Token, die nur aus dem Zeichen „`~`" bestehen, direkt nach dem „Zerlegen" durch den Heimatkatalog, wie er in der Umgebungsvariablen `HOME` zu finden ist.

Beispiel:

```
turing$ echo ~
/home/turing/jmayer
turing$
```

8.3 Variablen-Substitution (2 Punkte)

Ebenso wollen Sie, dass jedes Token der Form „`$Variablenname`" durch den Wert der jeweiligen Umgebungsvariablen ersetzt wird, sofern es eine Umgebungsvariable mit diesem Namen gibt. Andernfalls wird ein solches Token durch den leeren String ersetzt.

Beispiel:

```
turing$ echo $HOME $TTY $x
/home/turing/jmayer /dev/pts/85
turing$
```

8.4 Dateinamen-Expansion (2 Punkte)

Alles klar so weit? Dann wird's jetzt spannend! Das Token „`*`" wird von der Bash nämlich durch mehrere Token ersetzt, falls im aktuellen Katalog eine Datei (auch Verzeichnis!) existiert, die nicht mit einem Punkt („.") beginnt; nämlich für jede solche Datei ein eigenes Token. Gibt es keine solche Datei, dann bleibt das Token „`*`" unverändert.

Beispiel:

```
turing$ ls
midish.c midish test.txt
turing$ echo *
test.txt midish.c midish
turing$
```

8.5 Built-In-Kommandos (3 Punkte)

Sie haben bei der Midi-Shell auch mal `cd` gefolgt von `pwd` eingegeben. Tja, leider ist der aktuelle Katalog danach nicht Ihr Heimatkatalog. Da kommt es Ihnen! „Klar, für manche Kommandos darf nicht geforkt werden!" Erklären Sie das mal Ihrem Tutor anhand eines solchen Kommandos. Nun implementieren Sie die Kommandos `set` und `cd` als Funktionsaufruf (ohne zu forken!) in der Midi-Shell. So etwas nennt sich dann „Built-In-Kommando". `set` soll einfach die Umgebungsvariablen auf die Standardausgabe schreiben. `cd` soll bei einem Argument in das angegebene Verzeichnis wechseln oder in den Heimatkatalog, falls kein Argument angegeben ist. Bei Built-In-Kommandos soll in unserer Midi-Shell keine Hintergrundverarbeitung und keine Ein-/Ausgabeumlenkung stattfinden. (Wir machen uns das Leben leicht. :-)))

Beispiel:

```
turing$ set
http_proxy=http://www-proxy.uni-ulm.de:3128/
TMPDIR=/tmp
...
turing$
turing$ pwd
/home/turing/jmayer/html/feedback
turing$ cd
turing$ pwd
/home/turing/jmayer
turing$
```

Tipps: Auf die Umgebungsvariablen dürfen Sie jetzt natürlich mit Hilfe von `getenv()` zugreifen. Will man alle Dateien eines Katalogs aufspüren, sind die Funktionen `opendir()`, `readdir()` und `closedir()` nützlich. Und das aktuelle Verzeichnis wechselt man am besten mit `chdir()`.

Toi, toi, toi!