Übungen zu Systemnahe Programmierung in C

Abschnitt 12.4: Aufgabe (mish - Teil A)

13.07.2020

Tim Rheinfels
Benedict Herzog
Bernhard Heinloth

Lehrstuhl für Informatik 4
Friedrich-Alexander-Universität Erlangen-Nürnberg

Lehrstuhl für Verteilte Systeme
und Betriebssysteme

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

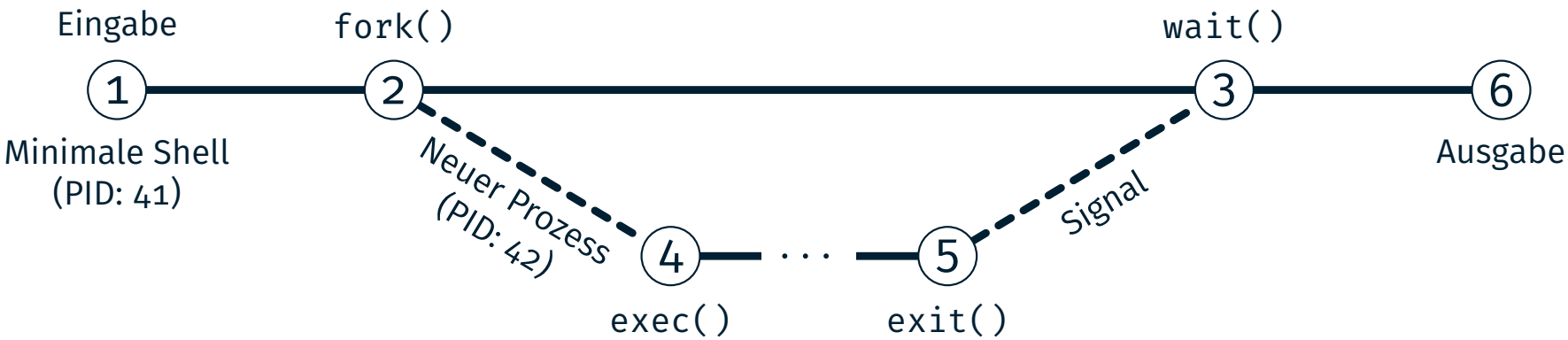
Aufgabe: `mish` - Teil a)

- Einfache Shell (**mi**ni **sh**ell) zum Ausführen von Kommandos
- Typischer Ablauf:
  - Ausgabe des Prompts
  - Warten auf Eingaben
  - Zerlegen der Eingaben
    - Kommandoname
    - Argumente
  - Neuen Prozess erstellen
    - **Vater:** Warten auf Terminierung des Kindes
    - **Kind:** Ausführen des Kommandos
  - Ausgabe des Exitstatus

Aufgabe: `mish` - Teil a)

- Wiederholung: Basisablauf einer minimalen Shell

1. Auf Eingaben vom Benutzer warten
2. Neuen Prozess erzeugen
3. Vater: Wartet auf die Beendigung des Kindes
4. Kind: Startet Programm
5. Kind: Programm terminiert
6. Vater: Ausgabe der Kindzustands



Aufgabe: mish - Teil a)

Beispiele:

```
01 # Reguläre Beendigung durch Exit (Exitstatus = 0)
02 mish> ls -l
03 ...
04 Exit status [2110] = 0
05
06 # Ungültige/Leere Eingaben
07 mish>
08 mish> foo
09 foo: No such file or directory
10 Exit status [7342] = 1
11
12 # Beendigung durch Signal (hier SIGINT = 2)
13 mish> sleep 10
14 Signal [1302] = 2
```

Einschub: `fflush()`

- Prompt druckt kein `'\n'`
- Standardbibliothek puffert `stdout` zeilenweise

⇒ Nach Ausgabe den Zeilenpuffers mittels `fflush(3)` leeren

Testprogramme

- Testprogramme: `/proj/i4spic/<idm>/pub/aufgabe8/`
- `spic-wait` (ohne Parameter)

```
01 mish> /proj/i4spic/[...]/spic-wait
02 [...]
03 - send 'SIGPIPE' to this process
04   Command: kill -PIPE 3372
05   Expected Output: Signal [3372] = 13
06
07 [...]
08
09 Signal [3372] = 13
10 mish>
```

```
01
02
03
04
05
06
07
08 $> kill -PIPE 3372
09
10
```

- `spic-wait (mit Parameter)`

```
01 mish> /proj/i4spic/<idm>/pub/aufgabe8/spic-wait 15
02 Sending signal 15 (Terminated) to myself (PID: 4239)
03 Signal [4239] = 15
04 mish>
```

Testprogramme

- Testprogramme: `/proj/i4spic/<idm>/pub/aufgabe8/`
- `spic-wait` (ohne Parameter)

```
01 mish> /proj/i4spic/[...]/spic-wait
02 [...]
03 - send 'SIGPIPE' to this process
04   Command: kill -PIPE 3372
05   Expected Output: Signal [3372] = 13
06
07 [...]
08
09 Signal [3372] = 13
10 mish>
```

```
01
02
03
04
05
06
07
08 $> kill -PIPE 3372
09
10
```

- `spic-wait (mit Parameter)`

```
01 mish> /proj/i4spic/<idm>/pub/aufgabe8/spic-wait 15
02 Sending signal 15 (Terminated) to myself (PID: 4239)
03 Signal [4239] = 15
04 mish>
```

Testprogramme

- Testprogramme: `/proj/i4spic/<idm>/pub/aufgabe8/`
- `spic-wait` (ohne Parameter)

```
01 mish> /proj/i4spic/[...]/spic-wait
02 [...]
03 - send 'SIGPIPE' to this process
04   Command: kill -PIPE 3372
05   Expected Output: Signal [3372] = 13
06
07 [...]
08
09 Signal [3372] = 13
10 mish>
```

```
01
02
03
04
05
06
07
08 $> kill -PIPE 3372
09
10
```

- `spic-wait (mit Parameter)`

```
01 mish> /proj/i4spic/<idm>/pub/aufgabe8/spic-wait 15
02 Sending signal 15 (Terminated) to myself (PID: 4239)
03 Signal [4239] = 15
04 mish>
```

Testprogramme

- `spic-exit`

```
01 mish> /proj/i4spic/<idm>/pub/aufgabe8/spic-exit 12
02 Exiting with status 12
03 Exit status [6272] = 12
04 mish>
```

Vorlage

```
01 // DESCRIPTION:
02 //   printStatus() examines the termination of a process and
03 //   prints the source of the exit (signal or exit) and the
04 //   exit code or signal number, respectively.
05 //
06 // PARAMETER:
07 //   pid:    PID of the exited child process
08 //   status: Status bits as retrieved from waitpid(2)
09 //
10 static void printStatus(pid_t pid, int status) {
11     // TODO IMPLEMENT
12 }
```

- `/proj/i4spic/<idm>/pub/aufgabe8/mish_vorlage.c`
- Die Vorlage enthält jedoch **nicht**:
  - Alle Funktionen, Funktionalitätsbeschreibungen, Variablen etc.
- Vorlage ersetzt nicht eigenständiges Nachdenken zur Struktur
- Während der Entwicklung kann es sinnvoll sein, das `Werror` Flag im Makefile wegzulassen