Technische Informatik 2 WS 2016/17 C04
Tutor: Marc Hildebrandt/ Bingbin Yu

Michael Schmidt
Stanislav Telis
Dominique Schulz
Norman Lipkow

Übungsblatt 4

1	2	3	4	5	$\sum$

Lösungsvorschlag
Abgabe: 28.11.2016

Aufgabe 1

Wir haben das vorgegebene Gerüst an den vorgesehenen Stellen um die geforderten Funktionen erweitert. Dabei haben wir eine Funktion mit dem Namen `process()` imlementiert, die die `PATH` Variable mittels `strtok()` in einzelne Pfade zerteilt, in denen dann nach dem auszuführenden Kommando gesucht wird. Das Suchen funktioniert mit der Systembibliotheksfunktionen `opendir()` für das Öffnen eines Verzeichnisses und `readdir()` für das Auslesen der einzelnen Dateien (bzw. die Programme die in der Environment Umgebung liegen). Wenn eine ausgelesene Datei mit dem Kommando übereinstimmt wird `excev()` aufgerufen und das Kommando ausgeführt. Darüber hinaus haben wir auch das `cd` Kommando mit der Bibliotheksfunktion `chdir()` realisiert.

Außerdem haben wir eine Funktion `check_child_status` implementiert, die mit Hilfe der Systembibliotheksfunktion `waitpid()` und der Option `WNOHANG` alle Kinder einsammelt und diese nicht nach der Termination in den Zombie-Zustand bleiben lässt.

Die `main()`-Funktion haben wir dahingehen erweitert, dass mit `fork()` ein Kindprozess des aktuellen Prozesses erstellt wird und in einer if Anweisung dem Kindprozess die Ausführung der Funktion `process()` überlassen wird. Der Vaterprozess wartet je nachdem, ob der Benutzer ein `&` Zeichen in die Kommando-Zeile eingetippt hat mittels `wait()` auf sein Kindprozess oder er wartet nicht.

Im Folgenden nochmal unser Code eingebunden:

```
1   #include <stddef.h>
2   #include <stdio.h>
3   #include <stdlib.h>
4   #include <string.h>
5   #include <sys/wait.h>
6   #include <unistd.h>
7   #include <iostream>
8   #include <dirent.h>
9
10  #include "parser.h"
11
12  using namespace std;
13
14  /**
15   * In dieser Funktion wird die Umgebungsvariable PATH mittels strtok()
16   * in Token aufgeteilt um anschließend in den einzelnen Verzeichnissen
17   * nach dem auszuführenden Kommando zu suchen.
```

```
18   * Wenn das Kommando gefunden wurde wird es mittels der Bibliotheksfunktion
19   * excev() ausgeführt.
20   *
21   */
22  void process(struct command c) {
23            struct command cmd = c;
24            if (strcmp(cmd.argv[0], "cd") == 0) {//kommando cd -> Bibliotheksfunktion
25                                                  //chdir() um Arbeitsverzeichnis zu wechsel
26              if ( chdir(cmd.argv[1] ? cmd.argv[1] : "") == 0) {
27              return;
28            } else {
29              perror("cd failed");
30              return;
31            }
32          }
33            DIR *directory;
34            struct dirent *file;
35            char *env = getenv("PATH");
36            if (env == NULL) {
37              perror("getenv() failed");
38              exit(1);
39            }
40            char delim = ':';
41            char *paths = strtok(env, &delim);
42            bool executed = false;
43            while ( paths != NULL ) {
44              directory = opendir(paths);
45
46              while ( (file = readdir(directory)) != NULL) {
47                if (strcmp(file->d_name, cmd.argv[0]) == 0) {
48                  char slash = '/';
49                  strncat(paths, &slash, 1);
50                  strcat(paths, cmd.argv[0]);
51                  if ( execv(paths, cmd.argv) == -1) {// Kommando ausführen
52                    perror("excev() failed");
53                    exit(1);
54                  }
55                  executed = true;
56                  break;
57                }
58
59              }
60              if (executed) {
61                if ( closedir(directory) == -1) {
62                  perror("closedir() failed");
63                  exit(1);
64                }
65                break;
66              }
67              paths = strtok(NULL, &delim);
```

```
68            }
69  }
70
71  /**
72   * Diese Funktion prüft den Zustand der Kindprozesse und sammelt diese ein, wenn sie
73   * terminiert sind. Dies geschieht mittels der Systembibliotheksfunktion waitpid() und
74   * der Option WNOHANG (sammle irgendeinen Kindprozess ein).
75   */
76  void check_child_status() {
77    while (true) {
78      pid_t pid = 0;
79
80      pid = waitpid(-1, 0, WNOHANG);
81
82      if (pid <= 0) {
83        break;
84      }
85    }
86
87  }
88
89  int main(){
90      for (;;){
91          struct command cmd = read_command_line();
92
93          if (strcmp(cmd.argv[0], "exit") == 0) {//Wenn exit eingetippt wird,
94                                                 //beende das Programm
95            exit(0);
96          }
97
98          cout << "command: " << cmd.argv[0]
99                    << ", background: " << (cmd.background ? "ja" : "nein") << endl;
100
101         check_child_status(); //sammle Kindprozesse ein
102
103         pid_t pid = fork(); //Kindprozess erzeugen
104
105         if (pid == 0) { //Kindprozess
106
107           process(cmd);
108
109
110         } else if (pid > 0) { //Vaterprozess
111
112           if (cmd.background) { //Wenn Kommando in Vordergrund -> Warte auf
113                                  //den Kindprozess (Shell blockieren), sonst
114                                 //warte nicht (Shell wird nicht blockiert)
115             continue;
116           } else {
117             wait(0);
```

```
118             }
119
120
121           } else { //Fehlerbehandlung
122             perror("fork() Failed");
123             exit(1);
124           }
125       }
126          return 0;
127    }
128
```

Tests

Wir haben das Programm auf dem x11 Rechner im Rechnerpool des MZH mittels `make` übersetzt. Dabei wurden uns keine Fehler angezeigt.

```
1  x11->make
2  g++ -g -Wall -Wextra -std=c++0x   -c -o ti2sh.o ti2sh.cc
3  lex  -t r.l > r.c
4  gcc    -c -o r.o r.c
5  g++ -g -Wall -Wextra -std=c++0x    ti2sh.o r.o parser.h   -o ti2sh
6  rm r.c
```

Anschließend haben wir unser Programm mittels ssh Verbindung auf dem x11 Rechner getestet. Dabei haben wir verschiedene Eingaben vorgenommen. Unteranderem das Starten eines Prozesses im Hintergrund, das Aufrufen eines Kommandos mit absolutem und relativem Pfad oder aber das Wechseln eines Verzeichnisses mit `cd`.

```
1  do_sc@x11 /home/do_sc/TI_2/uebung04/aufgabe1
2  ->./ti2sh
3  ti2sh$ ls
4  command: ls, background: nein
5  Makefile  parser.h  r.l  r.o  ti2sh  ti2sh.cc  ti2sh.cc~  ti2sh.o
6  ti2sh$ ls /home/do_sc
7  command: ls, background: nein
8  Desktop    Downloads  Linux  Pictures  TI_2       Videos    ausgabe  test
9  Documents  Library    Music  Public    Templates  aufgabe2  git      workspace
10 ti2sh$ cd ..
11 command: cd, background: nein
12 ti2sh$ ps
13 command: ps, background: nein
14   PID TTY          TIME CMD
15 11373 pts/3    00:00:00 bash
16 24059 pts/3    00:00:00 ti2sh
17 24112 pts/3    00:00:00 ti2sh
18 24114 pts/3    00:00:00 ps
19 ti2sh$ sleep 10 &
20 command: sleep, background: ja
21 ti2sh$ ps
22 command: ps, background: nein
23   PID TTY          TIME CMD
24 11373 pts/3    00:00:00 bash
```

```
24059 pts/3    00:00:00 ti2sh
24112 pts/3    00:00:00 ti2sh
24196 pts/3    00:00:00 ps
ti2sh$ sleep 10 &
command: sleep, background: ja
ti2sh$ ps
command: ps, background: nein
  PID TTY          TIME CMD
11373 pts/3    00:00:00 bash
24059 pts/3    00:00:00 ti2sh
24112 pts/3    00:00:00 ti2sh
24213 pts/3    00:00:00 sleep
24216 pts/3    00:00:00 ps
ti2sh$ kill 24213
command: kill, background: nein
ti2sh$ ps
command: ps, background: nein
  PID TTY          TIME CMD
11373 pts/3    00:00:00 bash
24059 pts/3    00:00:00 ti2sh
24112 pts/3    00:00:00 ti2sh
24244 pts/3    00:00:00 ps
ti2sh$ echo Hallo World
command: echo, background: nein
Hallo World
ti2sh$ ping -c 3 www.rocketbeans.tv
command: ping, background: nein
PING www.rocketbeans.tv (104.25.75.102) 56(84) bytes of data.
64 bytes from 104.25.75.102: icmp_req=1 ttl=59 time=7.46 ms
64 bytes from 104.25.75.102: icmp_req=2 ttl=59 time=7.43 ms
64 bytes from 104.25.75.102: icmp_req=3 ttl=59 time=7.45 ms

--- www.rocketbeans.tv ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 7.437/7.454/7.468/0.071 ms
ti2sh$ cat /home/do_sc/test/child1/u2/Makefile
command: cat, background: nein
CC=g++
CXXFLAGS=-g -Wall -Wextra
CXXFLAGS+=-std=c++11

myfind:: filter.o myfind.o
all:     myfind

clean:
        -rm -f myfind *.o core
ti2sh$ cd /home/do_sc/test/child1/u2
command: cd, background: nein
ti2sh$ cat Makefile
command: cat, background: nein
```

```
CC=g++
CXXFLAGS=-g -Wall -Wextra
CXXFLAGS+=-std=c++11

myfind:: filter.o myfind.o
all:     myfind

clean:
        -rm -f myfind *.o core
ti2sh$ ls -l
command: ls, background: nein
total 374
-rw-r--r-- 1 do_sc do_sc    127 Oct 25 18:01 Makefile
-rw-r--r-- 1 do_sc do_sc   3361 Nov  6 11:46 filter.cc
-rw-r--r-- 1 do_sc do_sc   1365 Oct 25 18:05 filter.hh
-rw-r--r-- 1 do_sc do_sc 488568 Nov  7 10:12 filter.o
-rwxr-xr-x 1 do_sc do_sc 277467 Nov  7 10:38 myfind
-rw-r--r-- 1 do_sc do_sc   2417 Nov  7 10:38 myfind.cc
-rw-r--r-- 1 do_sc do_sc   2370 Nov  6 10:24 myfind.cc~
-rw-r--r-- 1 do_sc do_sc  94072 Nov  7 10:38 myfind.o
ti2sh$ ls -l /home
command: ls, background: nein
total 31
drwx------ 30 chr_paw  chr_paw   41 Nov 28 17:29 chr_paw
drwx--x--x 60 claras   stud     285 Nov 28 13:53 claras
drwx------ 48 do_sc    do_sc     66 Nov 28 11:52 do_sc
drwx------ 44 janmohr  janmohr   61 Oct 12 12:44 janmohr
drwx------ 44 mehrfard mehrfard  73 Nov 28 15:44 mehrfard
ti2sh$ sleep 30 &
command: sleep, background: ja
ti2sh$ ps
command: ps, background: nein
  PID TTY          TIME CMD
11373 pts/3    00:00:00 bash
24059 pts/3    00:00:00 ti2sh
24112 pts/3    00:00:00 ti2sh
24428 pts/3    00:00:00 ti2sh
24583 pts/3    00:00:00 sleep
24589 pts/3    00:00:00 ps
ti2sh$ bla/fasel
command: bla/fasel, background: nein
ti2sh$ adkd
command: adkd, background: nein
ti2sh$ ls
command: ls, background: nein
Makefile   filter.hh  myfind     myfind.cc~
filter.cc  filter.o   myfind.cc  myfind.o
ti2sh$ ls -l
command: ls, background: nein
total 374
```

```
-rw-r--r-- 1 do_sc do_sc    127 Oct 25 18:01 Makefile
-rw-r--r-- 1 do_sc do_sc   3361 Nov  6 11:46 filter.cc
-rw-r--r-- 1 do_sc do_sc   1365 Oct 25 18:05 filter.hh
-rw-r--r-- 1 do_sc do_sc 488568 Nov  7 10:12 filter.o
-rwxr-xr-x 1 do_sc do_sc 277467 Nov  7 10:38 myfind
-rw-r--r-- 1 do_sc do_sc   2417 Nov  7 10:38 myfind.cc
-rw-r--r-- 1 do_sc do_sc   2370 Nov  6 10:24 myfind.cc~
-rw-r--r-- 1 do_sc do_sc  94072 Nov  7 10:38 myfind.o
ti2sh$ touch Makefile
command: touch, background: nein
ti2sh$ ls -l
command: ls, background: nein
total 374
-rw-r--r-- 1 do_sc do_sc    127 Nov 28 17:52 Makefile
-rw-r--r-- 1 do_sc do_sc   3361 Nov  6 11:46 filter.cc
-rw-r--r-- 1 do_sc do_sc   1365 Oct 25 18:05 filter.hh
-rw-r--r-- 1 do_sc do_sc 488568 Nov  7 10:12 filter.o
-rwxr-xr-x 1 do_sc do_sc 277467 Nov  7 10:38 myfind
-rw-r--r-- 1 do_sc do_sc   2417 Nov  7 10:38 myfind.cc
-rw-r--r-- 1 do_sc do_sc   2370 Nov  6 10:24 myfind.cc~
-rw-r--r-- 1 do_sc do_sc  94072 Nov  7 10:38 myfind.o
ti2sh$ touch filter.o &
command: touch, background: ja
ti2sh$ ls -l
command: ls, background: nein
total 374
-rw-r--r-- 1 do_sc do_sc    127 Nov 28 17:52 Makefile
-rw-r--r-- 1 do_sc do_sc   3361 Nov  6 11:46 filter.cc
-rw-r--r-- 1 do_sc do_sc   1365 Oct 25 18:05 filter.hh
-rw-r--r-- 1 do_sc do_sc 488568 Nov 28 17:53 filter.o
-rwxr-xr-x 1 do_sc do_sc 277467 Nov  7 10:38 myfind
-rw-r--r-- 1 do_sc do_sc   2417 Nov  7 10:38 myfind.cc
-rw-r--r-- 1 do_sc do_sc   2370 Nov  6 10:24 myfind.cc~
-rw-r--r-- 1 do_sc do_sc  94072 Nov  7 10:38 myfind.o
ti2sh$ ^C
do_sc@x11 /home/do_sc/TI_2/uebung04/aufgabe1
->./ti2sh
ti2sh$ cd ajddjn
command: cd, background: nein
cd failed: No such file or directory
ti2sh$ sdjkekcj
command: sdjkekcj, background: nein
ti2sh$ exit
do_sc@x11 /home/do_sc/TI_2/uebung04/aufgabe1
->
```

Aufgabe 2

Aufgabe 3