Technische Informatik 2 | WS 2016/17 | C/06
Tutor: Marc Hildebrandt

Niklas Koenen
Jan Klüver
Vincent Jankovic

Übungsblatt 4

1	2	3	4	5	$\sum$

Lösungsvorschlag
Abgabe: 28.11.2016

Aufgabe 1

Zuerst erkälren wir unseren Code der Shell und dann kommt ein kleiner Testbericht.

Code:

Nachdem der schon vorhandene Code durchlaufen wurde (eine if-Abfrage für *exit* wurde noch von uns hinzugefügt, denn dieser Befehl muss manuell angelegt werden), erzeugen wir einen Kindprozess mit *fork()*. *fork()* liefert uns die Adresse des Kindprozesses. Konnte der Elternprozess auf Grund eines Fehlers nicht dupliziert werden, gibt *fork()* eine -1 zurück. Diesen Fall fangen wir mit einer Fehlermeldung ab. Konnte der Kindprozess erzeugt werden, liefert *fork()* dem Kindprozess eine 0. Ist dies der Fall, wird der eingegebene *command* und die Paramerter mit der Envoirmentvariable *getenv("PATH")* am Doppelpunkt getrennt und in *sub* gespeichert. Vorher legen wir noch eine *bool* Variable an, die uns später Hilft, die Erzeugung eines neuen Prozesses zu überprüfen. Nach der Zerlegung wird mit einer *while*-Schleife jedes command (bzw. der Pfad) durchgegangen und umgewandelt. Zudem wird eine *stat*-Variable angelegt, die den *PATH* überprüft. Existiert der Pfad, so wird ein Prozess erzeugt und der command ausgeführt (mit *execv(s, cmd.argv)*). Danach sammeln wir alle Kindprozesse über den Signalhandler ein und benutzen *wait(0)*, um die Terminierung zu erkennen. Dann kann ein neuer Befehl eingegeben werden.

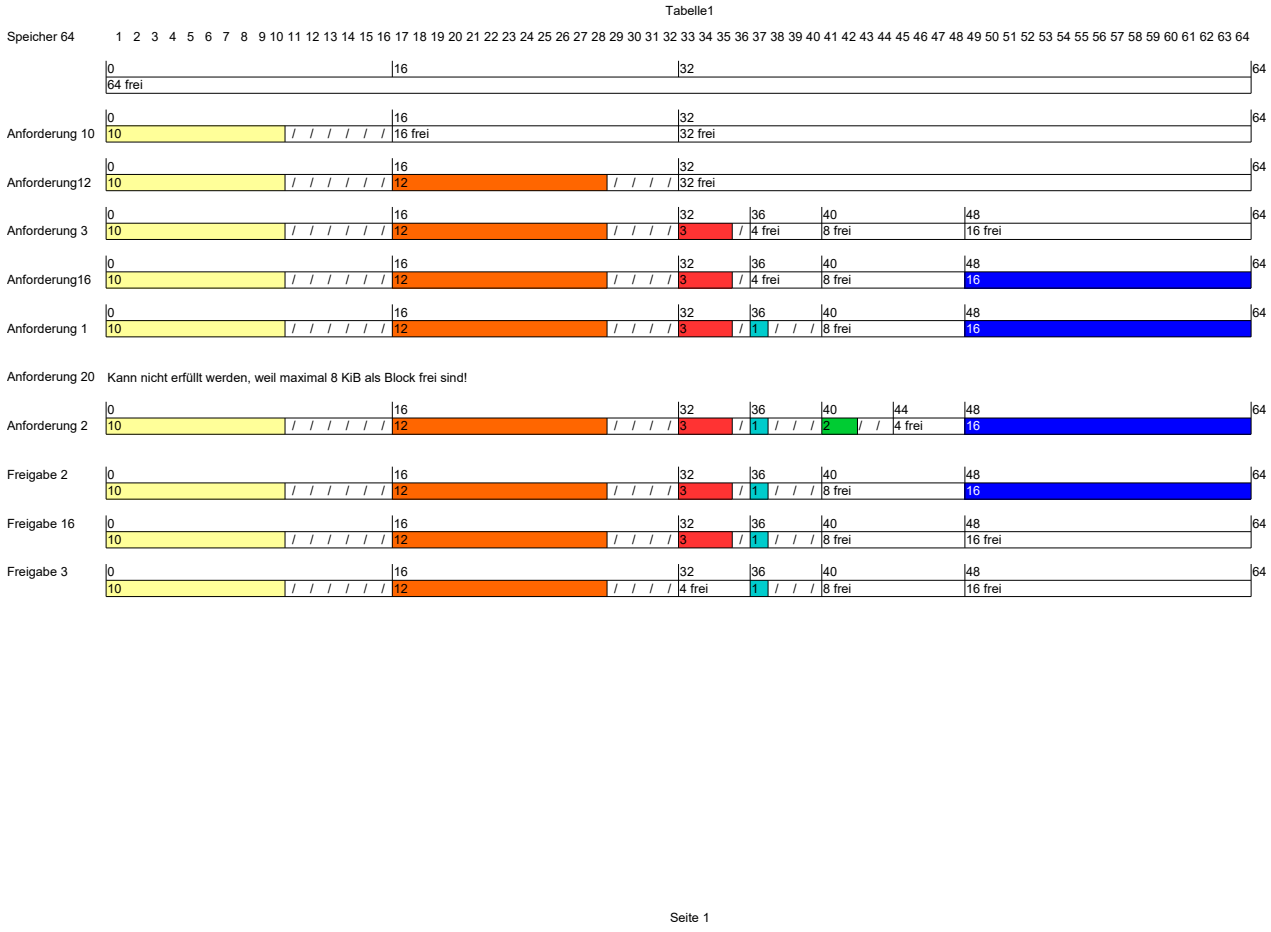
```
#include <stddef.h>
...

#include "parser.h"

using namespace std;

int main()
{
    for (;;)
      {
        struct command cmd = read_command_line();

        if(strcmp(cmd.argv[0], "exit") ==0)
          {
             exit(EXIT_SUCCESS);
          }
```

```
        cout << "command: " << cmd.argv[0]
             << ", background: " << (cmd.background ? "ja" : "nein") << endl;
        cout << endl;
        /* Hier muesste die Ausfuehrung stehen */

        pid_t pid;
        int status=0;
        bool neuerPro = false;

        pid=fork();

        if(pid == -1)
          {
            perror("Fehler!");
          }
        else if (pid == 0)
          {
            char* command = getenv("PATH");
            char* sub = strtok (command, ":");
            while(sub != NULL)
              {
                char s[200]= "";
                strcpy(s,sub);
                strcat(s,"/");
                strcat(s, cmd.argv[0]);
                struct stat st;
                if( stat(s, &st) == 0)
                  {
                    neuerPro = true;
                    execv(s, cmd.argv);
                  }
                sub = strtok (NULL, ":");
              }
            if(!neuerPro)
              {
                cout << "Sorry aber der Befehl ist Muell!!" << endl;
              }
            exit(EXIT_FAILURE);
          }
        else
          {
            pid=wait(&status);
            while(!WIFEXITED(status) && !WIFSIGNALED(status))
              {
                pid=wait(0);
              }
          }
    }
    return 1;
}
```

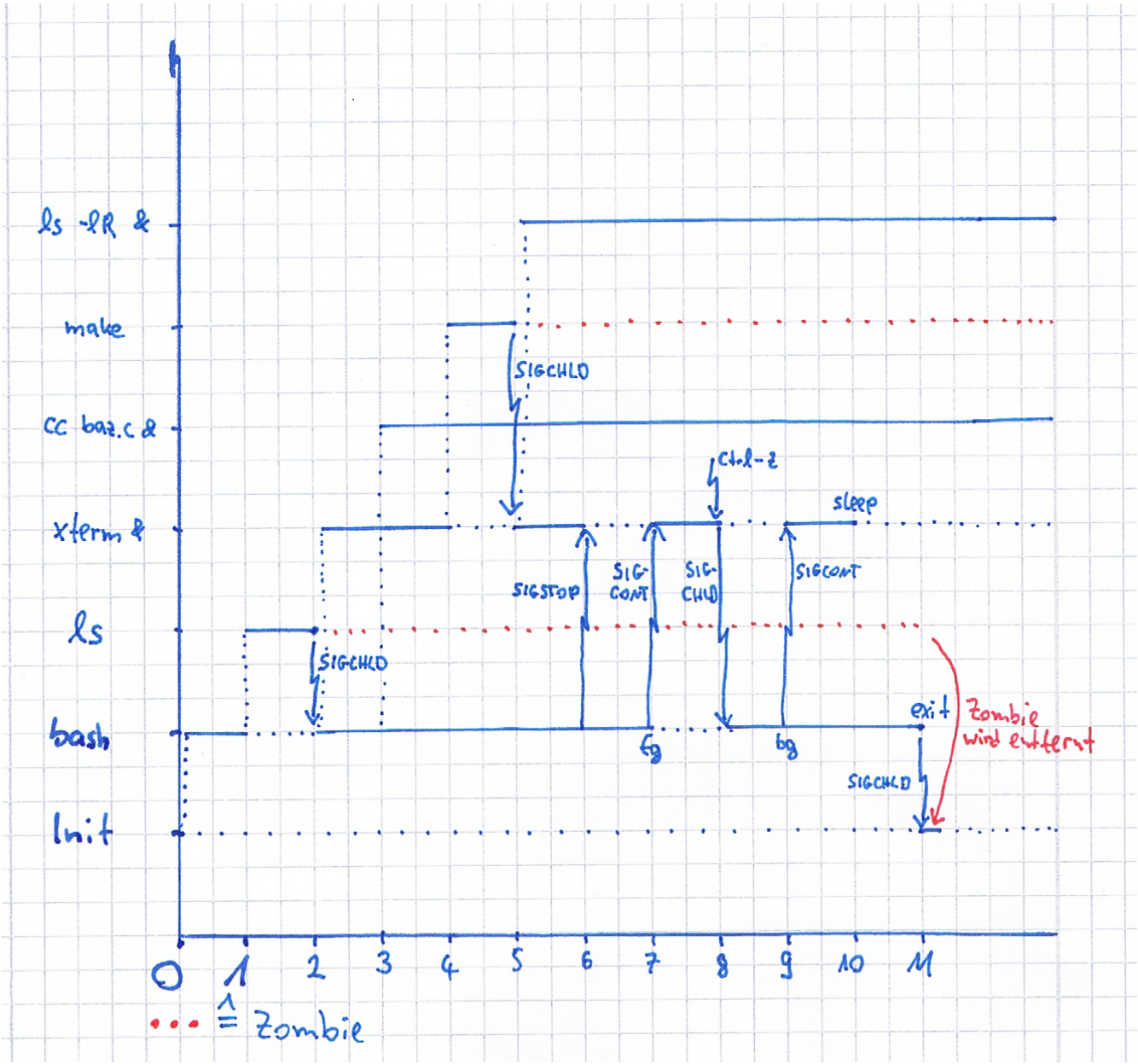
Testbericht:

Wir haben möglichst viele Eingaben getätigt, um unsere Shell zu testen. Am Anfang hat natürlich nicht alles geklappt bzw. kleine Fehler sind aufgetaucht, wie wir duch Tests feststellen konnten. Während wir unsere Shell erweitert haben, haben wird dies auch mit den Tests gemacht. Wir haben am Ende die Eingabetests, die im Tutorium angegeben wurden, ausgeführt, also eine unsinnige Eingabe, richtige Eingaben usw. Dabei haben wir festgestellt, dass *exit* nicht richtig behandelt wird. Dies haben wir manuell dann im Code geändert. Die Test Ergebnisse waren am Ende durchweg positiv.

Aufgabe 2



Aufgabe 3



Das gestartete Terminal müsste nun zu *Init* gehören, da der ursprüngliche Elternprozess *bash* terminiert ist.