Gruppe **20**
Betriebssysteme und Systemsoftware

Patrick Arens, 377536
Georg Manthey 376854
Benedikt Gerlach, 376944

Übung 0

Aufgabe 1.1

Aufgabe 1.2

a)

Sollen wir den Code hier auch reinschreiben? (Ans Löschen denken ;))

```
#include <stdio.h>
#include <unistd.h>

void child_A_proc()
{
  while (1) {
    fprintf(stdout, "%s", "A");
    fflush(stdout);
  }
}

void child_C_proc() {
  while(1) {
    fprintf(stdout, "%s", "C");
    fflush(stdout);
  }
}

void parent_proc()
{
  while (1) {
    write(1, "B", 1);
  }
}

int main(void)
{
  int child_A;

  child_A = fork();             /* neuen Prozess starten                    */
  if (child_A == 0)             /* Bin ich der Sohnprozess?                 */
    child_A_proc();             /* ... dann child-Funktion ausfuehren       */
  else {
    int child_C=fork();
    if (child_C == 0)
      child_C_proc();
```

```
37      else
38        parent_proc();              /* ... sonst parent-Funktion ausfuehren      */
39    }
40    return 0;
41  }
```

b)

Das Kommando `fflush` bei `fprintf` bewirkt, dass der Text aus dem Buffer ausgegeben wird. Da bei `fprintf` der auszugebende Text zunächst nur in den Buffer geschrieben wird, welcher erst ausgegeben wird, wenn er voll ist, kann es dazu kommen, dass auszugebender Text nicht ausgegeben wird, da der Buffer noch nicht voll ist. Also kann man den Buffer bereits vorher mit dem Befehl `fflush` ausgeben. Bei `write` wird der Text sofort ausgegeben, wobei intern auch zunächst der Text in einen Buffer geschrieben wird, welcher dann jedes mal direkt ausgegeben wird. Daher ist `fprintf` ganz allgemein auch effizienter, da das ausgeben der Buffers auf den Bildschirm auch Rechenzeit benötigt, was bei `fprintf` weniger oft gemacht werden kann.

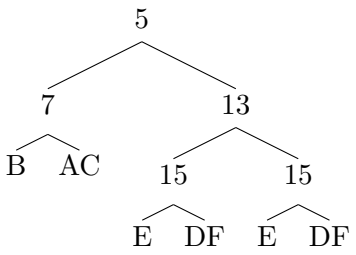
c)

Ein Zombie-Prozess ist ein Kindprozess, welcher beendet wurde, bei dem der Vaterprozess allerdings noch keine Statusabfrage durchgeführt hat. Dies erklärt nämlich auch den Grund für das existieren von Zombie Prozessen, sie geben dem Vaterprozess die Möglichkeit, herauszufinden, wie der Kindprozess terminiert hat (erfolgreich, mit Fehler,...). Der Zombieprozess verbraucht dabei kaum Arbeitsspeicher, da nur noch der Eintrag in der Prozesstabelle existiert.

Aufgabe 1.3

a)

Die Zahlen in den inneren Knoten stehen jeweils für die Zeile des aufgerufenen `fork()` Befehls.



b)

Angenommen, der linke Teilbaum wird als erstes abgearbeitet, kann man sagen, dass zunächst `A`, dann `B`, gefolgt von `C` ausgegeben wird, da der zweite Vaterprozess in Zeile 9 auf die Terminierung eines Kindes wartet. Über den rechten Teilbaum kann man sagen, dass auf jeden Fall ein D ausgegeben wird, bevor ein E oder F ausgegeben wird. Ansonsten ist die Reihenfolge unklar, da man nicht sagen kann, welcher Prozess zuerst ausgeführt wird.