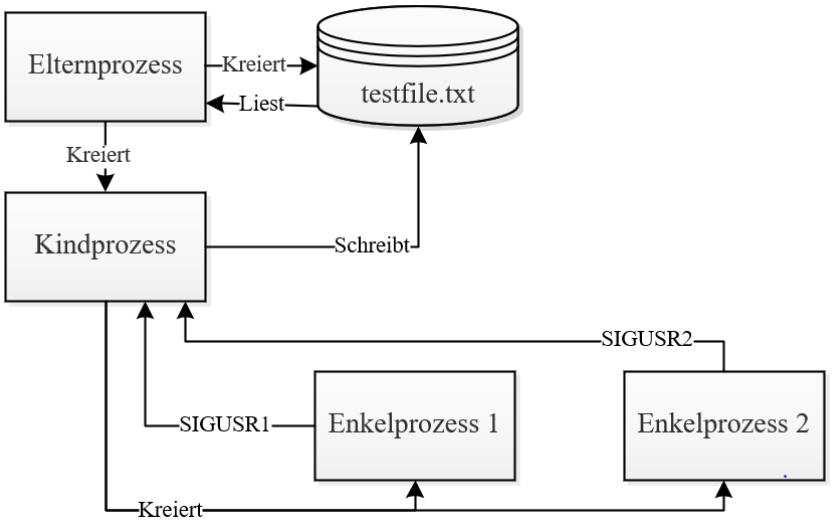
Stand: 17.03.2022

Aufgabe 2: Dateien, Prozesse und Signale

* **Ziele dieser Aufgabe sind:**
  - Kennenlernen der Arbeitsumgebung und der Werkzeuge für Linux/C
  - Kennenlernen einfacher Systemfunktionen (system calls)
  - Anwenden von Methoden der Fehlerbehandlung
  - Programmtechnischer Umgang mit Dateien, Prozessen und Signalen

* **Werkzeuge**
  Bei der Bearbeitung der Aufgabe 2 sollten Sie sich mit folgenden Werkzeugen vertraut machen:
  - Projektmanagement, Übersetzen, Laden
    `make`, `gcc` oder alternativ `Code::Blocks`
  - `gdb`, `kdbg`, o.ä.: Debugger, Programm schrittweise ausführen, Variablen betrachten und manipulieren
  - `ktop`, `htop` o.ä.: Listen der laufenden Prozesse und der Prozess-Zustände
  - Sequenzdiagramm

* **Überblick**
  Das Programm soll feststellen, welcher der Enkelprozesse 1 oder 2 in einem Zeitabschnitt wieviel Signale an den Kindprozess sendet. Dazu zählt der Kindprozess die von den Enkelprozessen gesendeten Signale und schreibt deren Anzahl in die Datei `testfile.txt`.



  - **`main`**
    Der Elternprozess kreiert die Datei `testfile.txt`. Dann kreiert er einen Kindprozess, der die Funktion `int kiddyscode(void)` ausführt. Danach wartet der Elternprozess auf das Ende des Kindprozesses. Dann liest er die Datei `testfile.txt` und gibt deren Inhalt auf der Konsole aus. Dann beendet sich der Elternprozess.
  - **`kiddyscode`**
    Der Kindprozess initialisiert passende Signalhandler für `SIGUSR1, SIGUSR2` und `SIGCHLD`. Der Kindprozess kreiert dann zwei Enkelprozesse, die alle die Funktion `int grandkiddyscode (int nr)` ausführen, wobei `nr` (1 …2) die Nummer des Enkelprozesses ist. Die PIDs der Enkelprozes-

se werden in einer globalen Variablen `int pids[2]` gespeichert. Die Anzahl der Enkelprozess wird in der globalen Variablen `int n_grandkiddies` gespeichert.
Wenn sich alle Enkelprozesse beendet haben, beendet `kiddyscode` den Kindprozess.

- **Signalhandler für `SIGUSR1` und `SIGUSR2`**
  Diese Signalhandler werden mit jedem empfangenen Signal `SIGUSR1` bzw. `SIGUSR2` aufgerufen. In den Signalhandlern wird mit jedem Aufruf eine globale Variable `countSIGUSR1` bzw `countSIGUSR2` inkrementiert und somit die Anzahl der gesendeten Signale gezählt.
- **Signalhandler für `SIGCHLD`**
  Der Signalhandler wird mit jedem Ende eines Enkelprozesses ausgeführt. Dabei wird jeweils `n_grandkiddies` aktualisiert.
  Im Signalhandlern werden nach Ende des letzten Enkelprozesses die Ergebnisse der Signalzählung aus `countSIGUSR1` und `countSIGUSR1` in die Datei testfile.txt geschrieben, wobei jeweils die PID des sendenden Enkelprozesse festzuhalten ist. Der Inhalt der Datei `testfile.txt` könnte beispielweise so aussehen:

```
Datei zur Speicherung der Signal-Hits
Enkel mit pid:8897 hat 52405-mal SIGUSR1 aufgerufen
Enkel mit pid:8898 hat 1378953-mal SIGUSR2 aufgerufen
```

- **`grandkiddyscode`**
  Zunächst initialisieren die Enkelprozesse einen Signalhandler für `SIGALARM`. Dann wird mit `alarm (MESSZEIT)` die Länge des Zeitintervalls eingestellt, während dessen die Enkelprozesse Signale an den Kindprozess senden. Anschließend senden die Enkelprozesse in möglichst schneller Folge Signale an den Kindprozess, wobei Enkelprozess1 `SIGUSR1` und Enkelprozess 2 `SIGUSR2` sendet.
- **Signalhandler für `SIGALARM`**
  Die Enkelprozesse führen diesen Signalhandler für `SIGALARM` aus, sobald `MESSZEIT` abgelaufen ist. Im Signalhandler beenden sich die Enkelprozesse.

*** Erweiterung des Programms**

- Sie werden feststellen, dass nach Ausführung des bisher vorgestellten Programms die Zählerstände `countSIGUSR1` und `countSIGUSR2` nahezu gleich sind. Das soll nun geändert werden.
- Versuchen Sie programmtechnisch die Prioritäten der Enkelprozesse zu Beginn von deren Ausführung auf unterschiedliche Prioritäten zu setzen, indem die Standard-Prioritäten erhöht oder erniedrigt werden. Wie verändern sich dadurch die Zählerstände `countSIGUSR1` und `countSIGUSR2`?
- Falls Ihnen das Verändern der Prioritäten nicht gelingt, überlegen Sie sich eine alternative Möglichkeit, um das Signalsenden der Enkelprozesse unterschiedlich schnell zu gestalten.

*** Bedingungen zur Lösung**

- Die Fehlerbehandlung von Systemcalls sollen i.d.R. mittels der Wrapperfunktionen realisiert werden.
- Die Programme sollen ausführlich getestet werden. Die Testausführung ist mit einer Beschreibung und mit Screenshots zu belegen.
- Es ist ein Sequenzdiagramm zum Ablauf des Programms (Anfang bis Ende) anzufertigen nach einem Muster in der Anlage.

*** Fragen**

- Wie zählt der Kindprozess die Anzahl der existierenden Enkelprozesse? Was ist für den Kindprozess das Kriterium sich zu beenden?

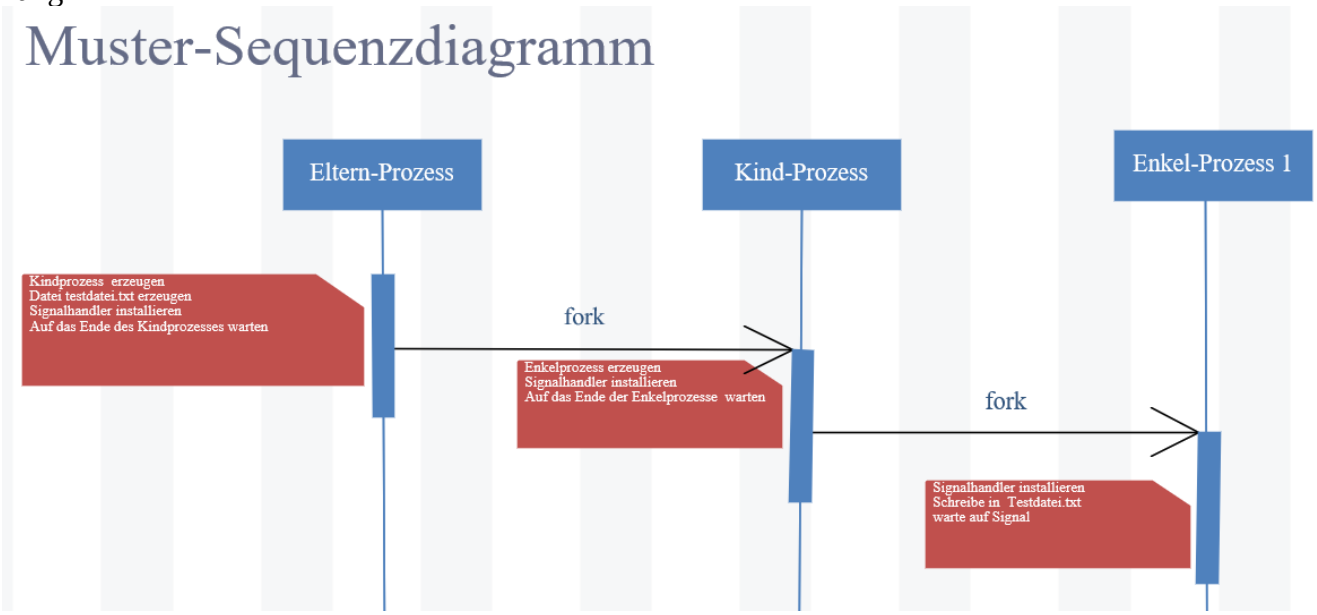
- Welche Prozesse/Programme brauchen Signalhandler für welche Signale?
- Welche Funktionalitäten müssen die jeweiligen Signalhandler enthalten?
- Wie lange dauert die komplette Ausführung des Programms vom Start bis zum Ende aller Prozesse?
- Überlegen Sie sich ein Konzept zur Erweiterung des Programms dergestalt, dass die Zahl der Enkelprozesse von zwei auf n (n ist z. B. gleich 4) erhöht wird, wobei alle Enkelprozesse das Signal `SUGUSR1` an den Kindprozess senden. Es soll in die Datei `testfile.txt` gespeichert werden, welcher Enkelprozess (erkennbar an der PID) wie oft ein Signal gesandt hat.
  - Was ist an den Funktionen `kiddyscode` und `grandkiddyscode` zu ändern?
  - Welche Signalhandler sind wie zu verändern?

* Abgabedokument

- Die Ausarbeitungen der Aufgabe muss in elektronischer Form in die ILIAS-Lernplattform eingereicht werden.
- Die Abgabe besteht aus dem Abgabedokument und aus den Programmen als gezippte Datei A2.zip.
- Das Abgabedokument enthält
  - Ein Inhaltsverzeichnis
  - Die Beantwortung der Fragen
  - Eine verbale Beschreibung der Vorgehensweise zur Lösung der Aufgaben
  - Die eingebundenen und kommentierten Programmcodes.
  - Die Dokumentation der Programmausführung (ist mit Screenshots zu belegen).
  - Das Sequenzdiagramm

- Dabei ist drauf zu achten, dass die Authentizität der Ausführung (Datum, Nutzerkonto) erkennbar ist. Eine Möglichkeit diesbezüglich besteht darin, den Prompt vor Ausführen des eigenen Programms entsprechend zu setzen:
- Das Abgabedokument soll im PDF-Format sein.

Anhang


Muster-Sequenzdiagramm
Eltern-Prozess
Kind-Prozess
Enkel-Prozess 1
Kindprozess erzeugen
Datei testdatei.txt erzeugen
Signalhandler installieren
Auf das Ende des Kindprozesses warten
fork
Enkelprozess erzeugen
Signalhandler installieren
Auf das Ende der Enkelprozesse warten
fork
Signalhandler installieren
Schreibe in Testdatei.txt
warte auf Signal